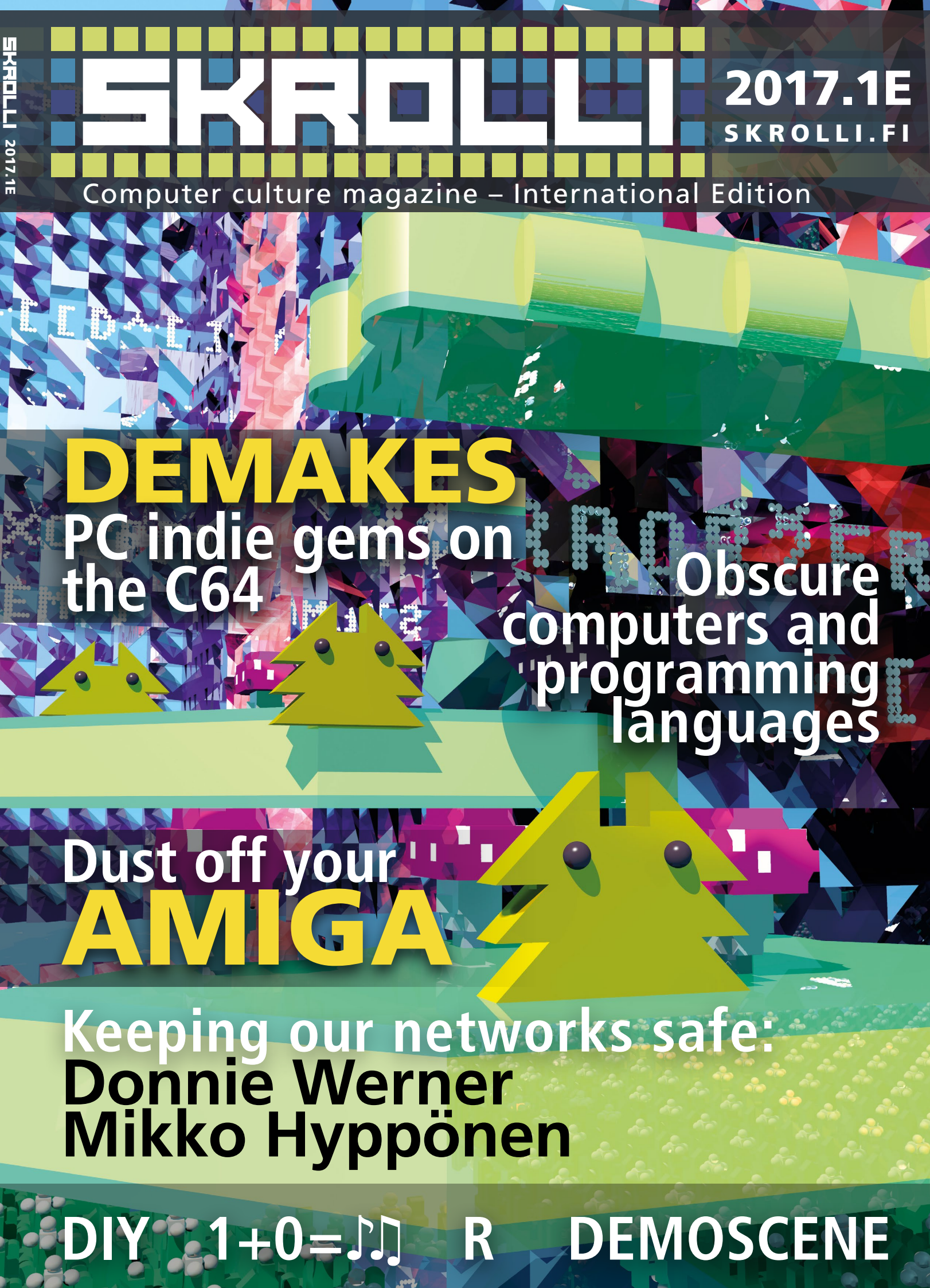
SKROLLI 2017.1E

SKROLLI

2017.1E
SKROLLI.FI

Computer culture magazine – International Edition



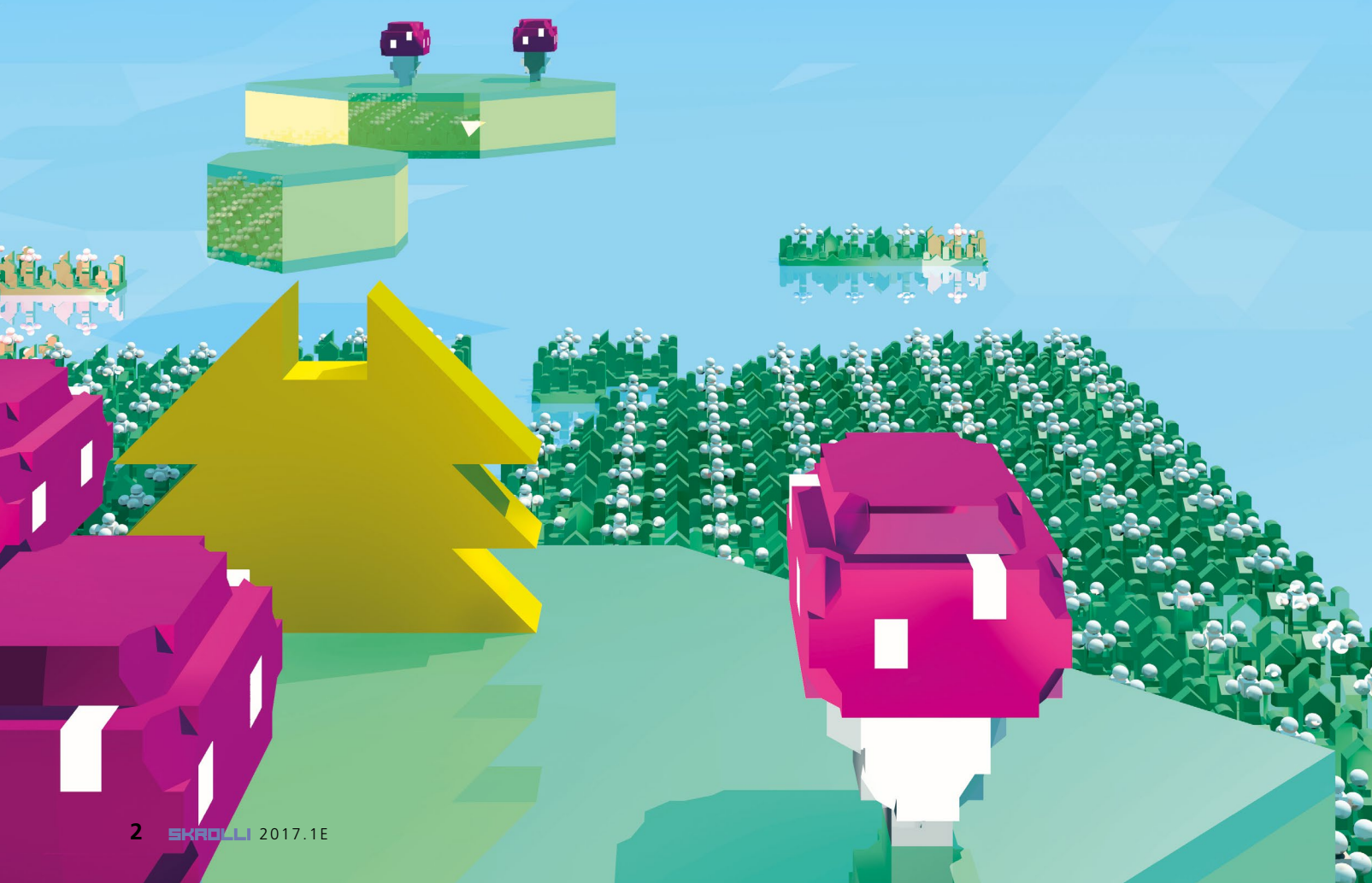
DEMAKES
PC indie gems on the C64

Obscure computers and programming languages

Dust off your
AMIGA

Keeping our networks safe:
Donnie Werner
Mikko Hyppönen

DIY 1+0=♪♫ R DEMOSCENE



Mikko Heinonen
Executive Editor

From Finland with love

Welcome to the second International Edition of Skrolli! Those of you who bought our previous issue (which is still available, by the way) will find that the composition of the stories is slightly different this time. For 2016.1E, we asked our IndieGoGo backers which stories from our back catalogue they would like to read, whereas this issue is a mix of our own favourites. Since the last issue, we have also had the honour of featuring some interesting people, such as **Mikko Hyppönen** and **Donnie Werner**, on our pages, and so we wanted to include those stories as well.

With this selection, we also hope to show that Skrolli is not simply a "retro mag". There are several great magazines on retrocomputing and gaming available in English already and so we have always felt that our mission extends beyond retelling about the past decades. We cover retro topics, but we are also a hacker magazine, a maker magazine and much more. We want to be a magazine for everyone with a deeper interest in computers, regardless of their age.

We decided to market this issue outside of the crowdfunding platforms in order to make it slightly cheaper for you. We have also created a lower-cost digital-only version that we hope will help Skrolli find new audiences beyond the fans of traditional paper magazines. Nevertheless, a hard copy version is available for those who want the real experience.

What the future holds for Skrolli, and these International Editions in particular, largely depends on you. If you enjoy reading these stories, please help spread the word about our magazine. In addition, we have not forgotten about the countless people who contacted us about the crowdfunding campaign and offered to help with creating content for the magazine. The fact that this issue, once again, consists almost exclusively of translated stories is due to time and resource constraints. If you wrote in and we do decide to make another International Edition, you will hear from us – and if you did not write in last year but feel like you could contribute, please get in touch!

Having read every story in this issue (and translated them, save for one), I can tell you that you are in for an exceptional treat. We have compiled something for everyone: for example, there's a great in-depth story about the Amiga 500, a hands-on USB joystick project, some statistical calculations in R and a cool model railway article. There is also some fear and loathing at a demo party, with a side order of sacred geometry. All of this (and more) is what Skrolli is about, and we would like to keep bringing the best bits to you in English in the future!

Skrolli
Computer culture magazine –
International Edition

Contact info@skrolli.fi
Ircnet: #skrolli
skrolli.fi/international

Executive Editor Mikko Heinonen
Editor-in-Chief Tapio Berschewsky
Managing Editor Valhe Kouneli
Art Director Nasu Viljanmaa
Digital Chiefs Toni Kuokkanen
Janne Sirén
Design and layout Manu Pärssinen
Image Editor Laura Pesola
Advertising sales Jari Jaanto
Finances Anssi Kolehmainen

Editors Jarno Niklas Alanko, Ville-Matias Heikkilä, Jukka O. Kauppinen, Ronja Koistinen, Ninnu Koskenalho, Sakari Lönn, Suvi Sivulainen.

Contributors to this issue Yrjö Fager, Marko Haarni, Mikko Happo, Chris Helenius, Ji Hyun Hong, Mikko Hyppönen, Matti Hämäläinen, Ville Jouppi, Jussi Kasurinen, Antti Kiuru, Toni Kortelahti, Jarno Lehtinen, Sakari Lehtonen, Sakari Leppä, Mitol Meerna, Heikki Mustonen, Jussi Paananen, Mikko Rasa, Markku Reunanen, Terho Tanskanen

Translation Mikko Heinonen

Proofreading Michael Harlan Lyman

Publisher Skrolli ry

Printed by Hämeen Kirjapaino, Tampere, Finland
ISSN 2323-8992 (print)
ISSN 2323-900X (online)

Cover image by
Ville-Matias Heikkilä

YMPÄRISTÖMERKKI
MILJÖMÄRKT
4041 0209
Painotuote
HÄMEEN KIRJAPAINO OY

My questions are too hard. When asked about the security of these cloud controlled smart home gadgets, the PR person isn't able to give out any reasonable answers.

Skrolli @ Techbrief
No common ground

"How can you make a profit with this? When you get home, contact me and we'll discuss some ads."

Story and translation by Tapio Berschewsky Images by Jukka O. Kauppinen

It's October 2016 and Skrolli is participating in a tech industry PR event in Stockholm, the capital of Sweden. I'm sitting at a table in the lounge of the conference centre, eating breakfast and introducing our magazine to a representative from a technology corporation when the question about ads is presented to me.

I'm hard pressed to come up with an answer. The non-commercial utopia that Skrolli represents is hard to explain in an event where every minute is optimised to make money for the participants. So why am I here then? To report how the industry works.

The invitation arrives

Techbrief is a relatively small regional event held annually in Sweden, where global corporations that produce consumer electronics and computer peripherals showcase to the various Nordic technology media outlets what's coming out next from their pipelines.

Even though Skrolli doesn't usually cover contemporary gadgets, we were asked to participate in the event. We told them we're not going to review the products.

Not a problem, they said. We could come by to report on the event itself. The only thing that the organisers asked from us is that if anything strikes our eye, we'd display the brands "in a Skrollish way". So we decided to go.

To begin with

So when the event finally arrived, I jumped out of my comfy bed way too early and took the train to the airport. I boarded the plane, closed my eyes for a bit, and suddenly Stockholm's Arlanda airport materialised around me. The event was held at the conference centre of the airport hotel, so reporters can get there and back again swiftly.

When I arrived, I was greeted with some breakfast. A croissant, some yoghurt, juice and coffee, and I was then ready to face the day properly. Let Techbrief begin.

For the next two hours, there were presentations from the bigger companies displaying their products at the event. During their allotted time, they had the chance to sway the attending crowd.

The audience was full of thirty-something representatives from various web publications, with a few younger, older and more print oriented exceptions in the mix.

The goal of each presentation is to get these often jaded and edgy people to pay attention to what's being said, and to coax them into writing about it later. The underlying hope is that their attention would convert into clicks by their readers, and those clicks would convert into purchases, which will lead to more flattering regional financial statements. Every minute counts.

The non-commercial utopia that Skrolli represents is hard to explain...

After just 15 minutes into the first demo, I was wondering what I was actually doing there. Skrolli is the antithesis of these kinds of presentations. The portion of our readers interested in new gaming mice or the latest external storage media for mobile phones and tablets do not look for information about them in our magazine. During the show, I saw

many products I might use personally, but hardly anything that struck me as professionally interesting today.

I started reminiscing about similar moments from the past decade. I've been on the other side of the fence many times, as one of these hungry editors looking for a scoop – or at least something that's interesting enough to publish. Trying to find the angle that would make my message spread further and faster than my rivals. To get those clicks, so my publisher can show more impressive figures for potential advertisers.

Each of the seven companies presenting have 20 minutes of time to conquer their audience. The timetable is strict and it shows. The presenters practically run through their talks and then in the end try to awkwardly cover what they forgot to say. In some cases, the presentations try to fit both a large number of new products and the vision and history of the company into a very short time. Some others focus on just a few select products and some jokes. The latter approach works better – they aren't as readily forgotten.

For Skrolli, all of this is foreign. To us, the two hours of demos are made up of noise with occasional observations on the differing styles of the talks. SanDisk tried to be funny but failed. Razer went after nostalgia and credibility by having a Commodore 64 and an Amiga 500 in their slides. Logitech involved us by taking us outside to test some wireless speakers. And then we had some lunch.

Intermission

After lunch, we were supposed to continue the day by mingling freely in a larger conference room, where each of the companies has their own stall for showing the upcoming products first hand to the reporters.

On top of the seven companies that were lucky enough to get a presentation slot, there's a few stalls for those who didn't get the chance. They have to make a lasting impression without podiums and PowerPoint slides. The tension is palpable.

I was not really surprised when – barely out of the presentation room and queueing for lunch – a representative of one of these unlucky companies clung to me and started talking about what they had brought with them. I promised to come by and have a look after some lunch. The PR guy smiles, nods, and starts talking to the next reporter in line. Behind that smile he looks pleading, maybe even a bit desperate.

After a short lunch, we continue towards the stalls.

Conversations

When you take into account that the sole objective of the event is to competitively push commercial messages into the channels of the attending media, Techbrief is gentle and light.

It's a really weird sensation to present your product to people who are actually there for the purpose of doing the same as you.

In huge global events, the same products are marketed with decibels that leave your ears ringing, dizzying lights, huge stalls that take architects to design, and booze fuelled afterparties laced with innuendo.

Only a few representatives stood behind small tables. Everyone had a few products to present. The only refreshments on offer were sodas. The atmosphere was very Nordic, a bit naive but businesslike and sober. I liked it there.

The conversations didn't really differ much from table to table. I was offered some products to test – which I refused. Everyone asked about Skrolli, but I was unable to present our concept in an understandable way. A technology medium that does not cover fashionable new gadgets or isn't very interested in commercial viability is a concept that was wholly unfamiliar to everyone else there. I'm a socialist in a room full of smiley gladhand capitalists. I could just as well have spoken a language that had been dead for millennia.

I handed out a few copies of our 2016 international edition magazine. People were puzzled and delighted. Especially the guys at the Benq stall liked what we're doing. It's a really weird sensation to present your product to people who are actually there for the purpose of doing the same as you. Especially since I can't sell them subscriptions, but only have freebies to offer. I had to explain the article we ran last year on Illuminatus many times to different people. Dated Finnish tech insider humour loses a lot in translation.

Eventually, I got a grip of myself and started doing what's actually expected of me. I began with the hard questions. For instance, how does the company demoing smart lighting systems and other smart home products defend them from hackers? Does the company do centralised firmware updates to patch uncovered security issues? What kind of precautions has the company taken? How secure are the products

Storage space grows at a slow steady pace, so it doesn't usually surprise me. However, holding the first memory card in the world capable of storing a terabyte of information felt pretty weird anyhow.

Razer and the demo effect. No matter how much he wanted to, the sympathetic guy at the Razer booth just wasn't able to demonstrate their nice looking new laptop – thanks to the update policy of Windows.

out of the box?

The representative smiled and told me I was full of good questions, but I could tell they really didn't get what I was saying. They claimed it's Amazon's responsibility to keep their cloud from being hacked. The possibility of a man in the middle type of attack between the user and the cloud or the cloud and the product goes straight over their heads.

I gave up. The people there were not experts at the technology they were peddling. They were just experts at peddling. They don't actually know what they're selling, but have learned the corporation's message by heart. For each and everyone there, that message was overwhelmingly positive. Everyone is the best and first at everything. And, I have to admit, I kind of liked some of the stuff I was seeing.

It was a fun feeling to hold in my hand a memory card that could store all the zeroes and ones that made up my 1990s. I also really liked the wireless speakers we were shown. They've come a long way in the 10 years I wasn't paying attention. And the small external storage units for mobile phones that SanDisk was presenting seem like something I might use.

It was at Razer's table where I found the most empathy. I discussed their gaming mice with the representative. I told him I've been using Deathadder models exclusively at home since their first release. We talked about some great old games. The representative turns out to be a former editor-in-chief for a large German gaming magazine.

Everyone asked about Skrolli, but I was unable to present our concept in an understandable way.

No wonder we had a lot of common ground. Even though the discussion was pleasant, I realised I was eating up a lot of his time, and decided to move on. I didn't want to impede anyone there, no matter how much fun we were having.

Spoils of war

Eventually, people started getting tired. The journalists stopped running around the room. The event organisers started handing out feedback forms. I couldn't answer any of the questions, since they all assumed I was there to familiarise myself with the products.

The reporters gathered in the first room where the presentations were held. Cameras and laptops were packed away with a wistful vibe. There was a lot of amiable but tired joking in the room. It was relaxing. Some of the people had hours to wait for their flights and would get home late at night. Others who left immediately teased them about it. It's friendly enough – most of the people had met before, and some were old colleagues.

People compared the swag they'd stashed away. Kingston branded plastic foam heads were abused. New gadgets were pulled out of their bags by chuckling people. The smiles revealed that, after the product reviews were done, these toys would not be returned. They'll remain on their shelves gathering dust. Foreign, but in some way theirs anyway. New, but soon forgotten.

At the end of the day, a few of the

Razer tried to woo the audience with nostalgic imagery. And hey, who doesn't like the Amiga?

Finnish journalists who were still there raised a glass together at the hotel bar. My head felt empty and heavy and it soon became oblivious what was going on around me. I drifted into my thoughts and replayed in my mind the conversation I had at breakfast.

A short flashback

I was sitting with another journalist at the breakfast table when a representative of a tech company sat down to get familiar with Skrolli. I tried to explain our concept.

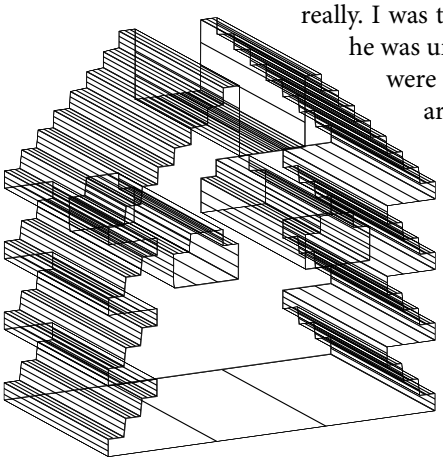
I told him we don't really write about products, but rather phenomena. No, not trends – we're interested in longer time scales. No, I don't mean stuff like the iPhone being kinda old now, even though the iPhone might be a good subject for an article in Skrolli if we get the angle right. I mentioned that we have a coding article for the ZX Spectrum in the latest Finnish issue of our magazine.

I showed him our international edition that was published in the spring of 2016. The PR guy started skimming through it. It took me a while to realise that he was looking for ads. I

I told him we don't really write about products, but rather phenomena. No, not trends – we're interested in longer time scales.

told him we don't really run many of them. The PR guy suggested we could run theirs and wanted me to contact him about it later. I mentioned we don't really want mainstream product ads in the magazine. We try to keep the number of ads to a minimum, and ensure all of them are relevant, so they won't annoy our readers.

Incredulity took over. He wanted to know how much it costs to subscribe to Skrolli. When I told him, he exclaimed that it can't be profitable. I revealed that our editorial staff works for free, and he looked at me as if I were a lunatic. I also mentioned that most of our freelance staff doesn't want to be paid for their content. Skrolli is a labour of love.



He wasn't listening to me anymore, not really. I was talking in a language that he was unfamiliar with. My words were not understood. There are no free lunches. He doubted me.

Suddenly, I stirred from these memories, and my glass was empty. I stood up, thanked my peers for their company, and walked briskly towards the gate and the kite that took me back to Finland.

The demo by Logitech was by far the most memorable. They gave each of us a wireless speaker, and we went outside to listen to some hits. It was kinda impressive having 20 or so wireless speakers playing music from a single smartphone. Nothing revolutionary, of course, but nice anyway.

That red foam head thing in the corner of the table. Kingston has been giving those out for years. And they've had to face petty abuse ever since.

Column

My week in information security

Mikko Hyppönen, Chief Research Officer, F-Secure

9 TUESDAY MAY 2017

In the morning, I drive to the F-Secure headquarters in Ruoholahti, Helsinki. I am looking forward to a few meetings and clearing my inbox. In the afternoon, I give a keynote speech at a cloud services seminar in a hotel in the centre of Helsinki. After scaring the cloud people, I return to the office to host a group of visitors in the laboratory. The Faraday cage is always a nice conversation piece.

10 WEDNESDAY MAY 2017

In the morning, I leave for the Expo and Convention Centre Messukeskus to give the opening speech at Process Days. The speech ends at 10:00, and in five minutes I am in a taxi, heading to the airport to catch the 11:05 to Oslo. One of the best Nordic hacker conferences, Paranoia, is starting there.

Wednesday's keynote speaker is Charlie Miller, the world's most famous car hacker. Me and Charlie go way back and we spend a long time talking in the afternoon. Charlie mentions that the vulnerabilities that he and Chris Valasek found cost Chrysler 14 billion dollars.

"If Chrysler had paid us 10 billion, we wouldn't have told anyone, and they would have saved 4 billion!"

After the conference, the speakers have dinner together. At least two former NSA employees sit at the table.

11 THURSDAY MAY 2017

I give the morning's first keynote at Paranoia. It is the tenth annual event. I talk about how much the world has changed over a short time. 10 years ago, each one of us had a Finnish phone in their pocket – now, none of us do. Time flies, especially on the Web.

After the session, I immediately open my Twitter since I know that F-Secure has published a stock market release: we are acquiring an English information security company. I spread the news on Twitter and welcome our new employees.

The large number of followers has taught me to be careful. Every now and then, I link to a site on Twitter and it immediately crashes due to too many visitors.

I manage my Twitter while in the taxi because at noon I fly to Stockholm for the IDG's annual IoT event. Speakers before me include the lady who designed Volvo's network connections and a man who connects reindeer to the Internet. I talk about the pros and cons of the future of IoT. My visit to Stockholm is brief. Immediately after my speech, I take the Arlanda Express to the airport and board the evening plane for Madrid.

12 FRIDAY MAY 2017

In Madrid, I speak to some 150 clients about corporate information security solutions. I have a few hours between my speech and the return flight. I return to the airport to do some work in the lounge.

I barely have time to sit down when my phone rings. Our office in Finland asks me to contact one of our major clients in Spain. I call a member of the board and find out that their network has thousands of computers infected by a new malware variant. What's worse, the infection is still spreading.

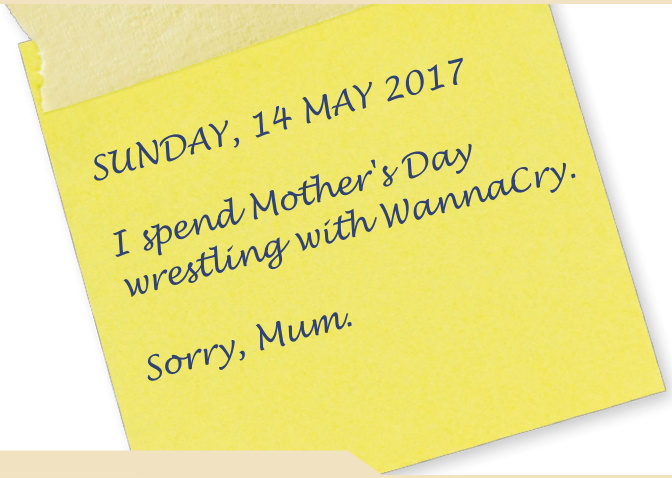
It is a new ransomware trojan – WannaCry. WannaCry is spreading like wildfire, and within one day it becomes the largest ransomware epidemic in history. It almost exclusively targets large corporations, and by the evening, there are nearly 200,000 infected machines.

My phone is ringing off the hook, and I even get in a few calls on board the plane as our departure is delayed. I have a stopover in Frankfurt in the middle of a thunderstorm. I still make the flight to Helsinki, since it is also delayed. I land at home base shortly before three in the morning on Saturday.

13 SATURDAY MAY 2017

I get less than six hours of sleep before the WannaCry circus continues. By the morning, our team has completely disassembled the WannaCry code. It turns out that WannaCry has stopped spreading, since a British researcher who I know personally found a function in the code that allowed him to stop the global epidemic. A deed worthy of a medal if there ever was one.

I do two live interviews with the BBC from my home over Skype. One of them is for BBC World that has dozens of millions of viewers. The interview goes well. There are infected companies all over the world: hospitals, car factories, power plants, train companies...



"My mother taught me to go wherever my work takes me."

15 MONDAY MAY 2017

On Monday, we find out that WannaCry did much more damage in Asia than expected. The infections in corporate networks were not detected before the weekend. The phone keeps ringing. I get constant calls from clients and the media.

I record a two-minute summary of the situation and respond to interview requests from radio stations by emailing them an OGG file. Radio stations from South Africa and Austria, at least, play the file to their listeners without edits.

CNN calls me and wants to interview me live over Skype for international distribution. The interviewer is a female journalist from Hong Kong whom I've met several times over the years. Before going live, we reminisce about interviews regarding the Sony Rootkit in 2005 and Conficker in 2008. After CNN, I take a taxi to Helsinki Airport.

As I write this, I'm sitting on board the Finnair flight to Barcelona. Tomorrow, we will be starting F-Secure's most important annual client event with operator clients from all over the world. It is not difficult to guess that WannaCry will be the number one topic.

From Barcelona, I'm headed to The Hague to discuss WannaCry with investigators from Europol.

As you now see, as an information security worker, I truly need to go where my work takes me.

Donnie Werner

From hacker to security consultant

"With what I know and what my friend knows – just the two of us – we could cripple the world."

Story by Valhe Kouneli Images by Mette Erikkilä, Valhe Kouneli, Nasu Viljanmaa

We're sitting at the corner table of the beer house Amsterdam in Ruoholahti, Helsinki with Donnie Werner. The rough looking man in his 50s is wearing a black leather jacket and talking about a "fire sale". This is a media term that refers to toppling the world's infrastructure, such as electricity distribution, communications and so on.

When Donnie was last in the US, he and his hacker friend had been talking about how long it would take to make something like that happen.

– We figured you give us two weeks for recon and then two days – BAM – it's done. We'd do a lot of damage.

Coming from most other people, this would sound cocky, but we already know his background, and don't need to question his story.

Donnie – and we're talking to Donnie instead of Mr Werner, since that would feel way too formal – is an old school hacker.

– There are a few critical spots in systems that you can hit and things go down. But I'm not malevolent.

Who is Donnie Werner?

These days, Donnie does penetration testing for a massive telecom corporation, protecting the global systems that we all take for granted. But back in the day he was a black hat hacker, breaking into systems all over the world from his keyboard.

In his youth, he spent his free time in a fairly unconventional manner. He tells us that, at weekends, he and his friend got together and chose a country and a government website to penetrate.

– It was like let's see what we can do today. We had many successful experiences. However, we never broke anything, we were not politically motivated.

The purpose of hacking was just meant to test their own skills.

Online, amongst his peers, he is known better by his alias: morning_wood. He also ended up in a Hacker Jeopardy trivia question on Defcon: "Who named Whoppix?"

– Whoppix was a predecessor to BackTrack Linux. It stands for Whitehat Knoppix.

The man behind the name

The recognition is deserved – Donnie's influence on the hacking subculture has been far reaching.

When the Remote Auditor and Slax projects that preceded BackTrack were combined, Donnie ended up among the authors. He credits his success as a hacker to getting the attention of other hackers at an early stage and meeting the right people.

"I was always the little kid who broke everything apart to see how it works."

In addition to Whoppix, Donnie also says he came up with the name for Whax, Whitehat Slax.

– Muts actually named BackTrack, but we did have an extensive conversation between each other about that. I was involved with it until just before Kali was released.

Muts is a nickname for **Mati Aharoni**, a hacker who founded Offensive-Security, a leading company offering offensive security training.

The accident

We buy Donnie a drink and ask more about his background. His stories go back to the California of his youth.

– I basically grew up on an airport and was involved in that type of work, mostly refilling aircraft. My father ran a line service. I was going to get a job in that industry. However, I was in a bad motorcycle accident in 1987, and that changed everything.

He points to his arm, which is covered in the sleeve of a black leather jacket and wouldn't otherwise attract attention.

– I lost the use of one of my arms, and could no longer do physical work such as property maintenance. I had to think of something else.

– I was always the little kid who broke everything apart to see how it works. As I got older, I was always fixing computers and other stuff. A career with computers aligned really well.

Donnie tells us how working at a computer repair shop made him service manager, and for a few years, he spent his days with repairs and diagnostics. During this time, he already started to encounter things that he later found were related to hacking and information security.

He also says that he was studying to become a mechanical engineer, but dropped out after a couple of years.

However, he says that the studies directed his thinking and improved his understanding of himself.

Early experiments

Although it took some time for Donnie to actually realise his potential as a hacker, the first time he acted like one was back in school.

– I was already hacking when I used a computer for the first time.

– I was in a beginning computer class, learning flowcharts, and I didn't really understand the goto loop. I could not wrap my mind around parallel processing, he says while tracing his finger across imaginary program code: – My brain was stuck.

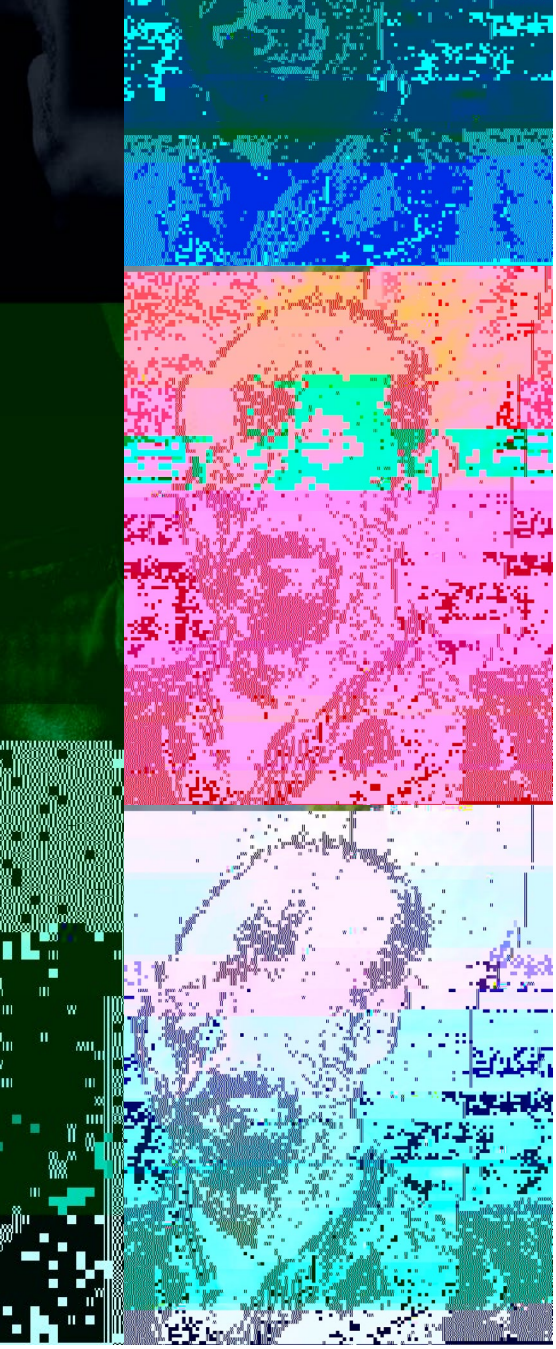
"My first computers were from dumpster diving."

Donnie says that the class had terminals that you used to log in with a server name and user ID. He noticed other names that looked like server names and could not resist the temptation. Finally, he got into the school's system.

And, of course, that was not the end of his hacking experiments. Soon, penetrating systems became a daily pastime.

– I started to discover the dial-up stuff and got into what was called *war dialling*. That's where you use an automated program to dial phone numbers for you. So whenever you would connect to another computer, it would record that communication, that handshaking, and whatever log-on was there.

– I would go to work and have it running all day long. Then, at night, I'd come back and explore the results and try to log in to the systems I found. Sometimes, you didn't even need to log in. I was exposed to a lot of different operating



systems.

His eyes take on an excited glimmer as he recounts his early hacking memories. You can feel the passion towards exploring new things – even when they are behind a stop sign – in his entire character.

Curiosity took over and the hacker started digging deeper – even quite literally. A lack of funds drove him to explore the dumpsters behind computer stores, and he says that his first computers were put together from parts he had found there.

He found hardware, software, user IDs and passwords – the latter were most often written or printed on paper. He also found books about operating systems.

– By chance, I would be exploring new systems, and some log-in – say, HP-UX and SCO, and then maybe a week later, in the dumpster, there'd be like a stack of SCO books!

Reading *Hacking Exposed* was perhaps a life-changing event for him. By the time he found it, he already had a broadband connection.

– I found it extremely interesting. They had things about HTML and how the Internet worked, but it also had a kind of inside information, almost like secret information in a way. I studied it carefully and tried out the techniques.

– I learned many basic techniques used in security work, like *reverse netcat*. It also explained the basis of connecting to another computer, using tools like *nmap* to discover what the other computer is running and things like that.

– Around the year 2000, I started to focus on security as an interest. I was discovering some things on the Internet, just through my browser, wondering how some things work, and then started to learn HTML and TCP/IP. I did it on my own.

First steps

His career as a hacker was far from sealed. Although he did not realise it at the time, it is possible to pinpoint when Donnie started gravitating towards hacking as a career.

While experimenting with a browser, he was able to reverse engineer the software of an adult web cam service. The system had severe vulnerabilities that allowed for collecting the models' IP addresses directly from the website.

– I kept that information private for a really long time, but then decided that the privacy of the models was of more importance. At that time, there were really bad exploits against Windows systems, so compromising their systems would have been really easy if you knew the IP address. It was really trivial to do so.

Donnie tried to contact the company directly and to offer consultation for a fee. His goal was to fix the vulnerability in the end. However, the company did not agree.

– So I went ahead and released the information to the public. That forced them to act, he states calmly.

Reputation

Donnie was now a hacker, and it didn't take long before he found others like him.

– I had met a group of hackers behind the Sub7 trojan, and I fell in with that crew pretty deeply. Many of the people that were involved are now long time close personal friends of mine. Some of them are top people in the industry as well, and their background is basically hacking people, using backdoors and hijacking computers, he says with some amusement.

> *"I was already hacking when I used a computer for the first time."*

Meeting other hackers also introduced Donnie to the principle of full disclosure and the *Full disclosure* mailing list, whose purpose is to force software vendors to fix problems and to make people aware of them. Donnie also wanted to do something about the problems he had found.

– First, I was just releasing things I had found without contacting the vendors or anyone else.

– As I found more and more problems, I decided that I wanted to start getting recognised for it. In a way, I felt that I would use this to build myself up, and I did. I was quite good at what I did.

Donnie gained a reputation and grew his résumé by finding vulnerabilities in products from giants such as HP, Microsoft and Apple. He also helped to correct the problems he found and got his name in the official updates. The increased reputation allowed him to start working as an independent security consultant, finally earning a living with his special skills.

On coming to Finland

– As a self-employed security consultant, I worked from gig to gig. Busting my ass for two weeks to get three to six thousand dollars – but never knowing when the next gig was coming. There might have been something lined up, but then something else happened, and all of a sudden I realised I hadn't worked in ninety days.

As the bills added up and there was constant uncertainty regarding the next assignment, he decided he no longer wanted to be an entrepreneur.

A few years prior to moving to Finland, he had done a series of classes in Tallinn involving hands-on hacking, seminars and lectures. This meant that

he was familiar with the area. When the opportunity for moving to Finland presented itself, he did not hesitate for long. This was despite the fact that he did not have a job yet.

– I only got a job after moving here. Coming to Finland as an immigrant, I felt that it was my responsibility to earn my own way, and to contribute to the society through my taxes.

– There were no openings, so I contacted companies directly. I went online and searched for "Helsinki computer security company". I sent my CV out to a few places, and got the first contact soon. Eventually, I ended up at an international telecom company.

At the time of writing, Donnie has been a security consultant for his current employer for about six years. After working at the product department and the security unit, he transferred to the group business unit where he conducts penetration testing, formability assessment and consultation.

Corporate-level idiocy

Although Donnie works full time as a security consultant for a big corporation now, in his free time he is still prodding away, learning new things and breaking stuff apart to find out how it works.

– Just recently, I discovered a flaw in an IoT product. The technique that I used is so absolutely stupid that I should have never tried it in the first place. It is an absolutely non-RFC compliant type of URL string that I've discovered provides an authorisation bypass to critical system files.

He explains that, by constructing the URL in a special way, in violation of RFC rules, he has been able to extract usernames, passwords and other things from the device without authentication.

Earlier, I had seen a somewhat similar technique used in a somewhat similar product, so I tried a variation of it. And it worked. I've been trying it in lots of other stuff and it simply isn't really working, so I understand that it's likely just to do with the implementation of URL handling in this particular product.

"You have an exploitable security issue."

Donnie will not tell us exactly what the device is, but he describes it as a personal, intimate product that people mostly use at home. He doesn't think anyone else has found the vulnerability yet. According to him, there are about half a million of these devices, and they are all online. One of the resellers is in the United States.

– I tried to contact them and explain what I had found. They asked 'what product is that' because they make a few different ones. So I described their product, and the reply after that was "*Hi. Here's the link to our product. Please check the product description for more information.*"

– They sent me a damn Amazon link, he sniffs and continues to read the email correspondence he has printed out.

The discussion with the reseller representatives was so absurd that he wanted to share it with his colleagues, and he still happened to have the printout with him.

– So I wrote back, and I said:

"*I think you did not understand. I am not looking for a specific product. I have discovered many security issues affecting all of your product(s). There's unrestricted access, privacy disclosure, and full control of your product.*

Most companies offer what is known as a bug bounty, which is preferable to the media attention caused by a disclosure of information. A public disclosure would jeopardise the privacy of the users and may lead to your product not being used.

Your product is designed to be used intimately and privately, and people hope to be able to trust it."

– So they replied: "*Many thanks for asking about our product, an easy to install product, where security is at the administrator's password settings, so no need to worry about security. It is secure to use and easy to install.*"

– I wrote back with "*Thank you, but you do not understand. Please show this email conversation to a technical manager.*"

– And the response was: "*Hi. As far as privacy and security goes, the possibility of others hacking into the product will increase if you use Wi-Fi. I suggest you use a password secured Wi-Fi connection, if possible. Next, after the set-up, I would go into the application and change the username and password.*

Upon set-up, it provides you with a generic username and password so that you can later go in and change it. You can do this to provide extra security. You can also go in and change the alias and name of the product so that people cannot see that it is this particular product."

– My last reply to them was: "*Sadly, you do not understand or care. You have an exploitable security issue. This is nothing to do with set-up, and this can be bypassed from the Internet without first authorisation.*"

There is disbelief and frustration in his voice, even though the correspondence took place some time ago. He says that he received one more message.

It was: "*I apologise, as we do not seem to understand where you are coming from. We have explained everything about our security measures. Additionally, our product cannot be accessed without the device ID which is uniquely designed.*"

Donnie says that he had penetrated the system a couple of months earlier, even though he had only acquired the device ID a couple of days ago. He does not spare his words when criticising the poor implementation of the product. However, publishing the information would lead tens of thousands of others to attack the device. Donnie cannot think of a reason to do so.

– The only motivation would be some type of financial remuneration. I would really like a bug bounty. The vendor obviously does not recognise that they even have a security flaw, and so they sure as hell aren't going to pay a bug bounty.

There is another reseller in Great Britain that Donnie might still try to contact. However, it is likely that the case remains open.

Unfortunately, this is not the only company to provide a substandard response to a security issue. Donnie says that gets a lot of similar responses; however, many companies also take bug discoveries seriously.

Favourite conspiracy

When Donnie is asked whether there is some favourite story about a security catastrophe, he doesn't hesitate at all: MS Blaster. Sharing this personal experience makes his eyes glow again.

– In 2003, there was this blackout in New York and the whole East Coast. They blamed it on fallen trees, train tracks and whatnot. When I heard it on the news, my first thought was: ha, blasted! He laughs.

– **Mikko Hyppönen** appears in the same documentary with me, and he openly says that he thinks this is what happened. Those outside the hacker community will not admit it, but we *know* that MS Blaster cut off the power.

– I don't believe there was any motivation behind the attack, he says. After the first proof-of-concepts of Blaster were done, they were turned into worms, and the worms only did IP ranges. The worm got an IP range from one machine, scanned it, expanded that IP range and then scanned that range. It did not select its targets in any way.

Donnie says that he met the worm at an early stage.

– Some of the people who discovered it or had access to the very early first round of it, actually used it, targeted people with it, and then used those people to attack me in a denial-of-service. I only found this out because I actually hacked them, he reveals with a hint of pride in his voice.

The worm only got its official name a couple of weeks later and its existence was recognised.

> ***"Considering security as an afterthought is not enough."***

Awareness

Rather than putting the blame for security breaches on the individual users, Donnie thinks security should be assessed from the development side.

– In security, education is extremely important. We need to increase awareness about security flaws, not only for developers but also for those who recruit developers. I think developers should be tested better in terms of security. Considering security as an afterthought is not enough. It should be a part of everything you do, he states.

Although Donnie is among the best in his field and can break others' code, he has never worked in software development himself. It gives him a unique perspective on the current state of se-

curity.

– I was struggling to get into the industry, and at the same time, I saw people that get paid a lot of money – and somebody like me comes along and just breaks their stuff. That, to me, should not happen. Someone made a bad decision trusting this guy. The management has hired a person who misrepresented themselves. A person who writes poor code, he exclaims, visibly irritated.

According to Donnie, developers cannot hide behind the fact that security was never taught at school, or that the intention was never to make the product secure. According to him, you should learn security as you are learning a language or development environment.

– If you've chosen to develop on a certain platform, you should be well versed! And not only in that; in everything else as well. Not only in development, but also in the security problems related to the development or the environment. That's what you get paid money for! That's why you went to school!

Confidence

Testing by developers is another topic that gets Donnie going.

– They test things like security issues, out of date security patches and bad permissions – that type of thing. But at the same time, they don't test parts of their web application or how they are handling input, and things like that. You should have an attitude of open exploration.

According to Donnie, this comes naturally to him, but there are also people who find it difficult. He describes his working methods as leaps into the unknown.

– Often when I do something and it causes an unexpected output or return, the developer says 'that's not how you use it'. Okay, well, that's great, but that's how I used it. And if I can use it that way, somebody else can use it that way as well. So you design around that.

– This is another thing that the developers do. They're developing a product; net product, some application – whatever – and it runs on top of an operating system. And oftentimes when you ask for the security test results back, ultimately what they give you is actually the testing of the operating system. They don't quite know how to tailor the tests or the tools towards their product. Because most of the tools in their base configurations just test the operating system, basically.

Regardless of what operating system he is using, Donnie prefers to use real machines instead of virtual ones.

– I just bought a used Windows tablet today for forty bucks. Why did I buy it? Because it has Windows. There are a couple of Windows programs that only work in Windows, and I would like to use them. I don't like virtual machines. I'd rather have ten laptops than one and nine virtual machines, Donnie sneers.

He explains that, for example, when testing what happens to packets on a network, using a virtual machine is problematic. When you send your packets from inside that virtual machine, they have to go through a virtual ethernet adapter. And when those data packets come back, they have to go through another virtual ethernet card again.

– If you're doing, let's say, low-level protocol fuzzing, how can you really understand the nature of your results if that virtual stack is not fully open and compliant like a regular physical hardware network card would be? So you're relying completely on the driver implementation of that virtualisation. It's not good enough for me.

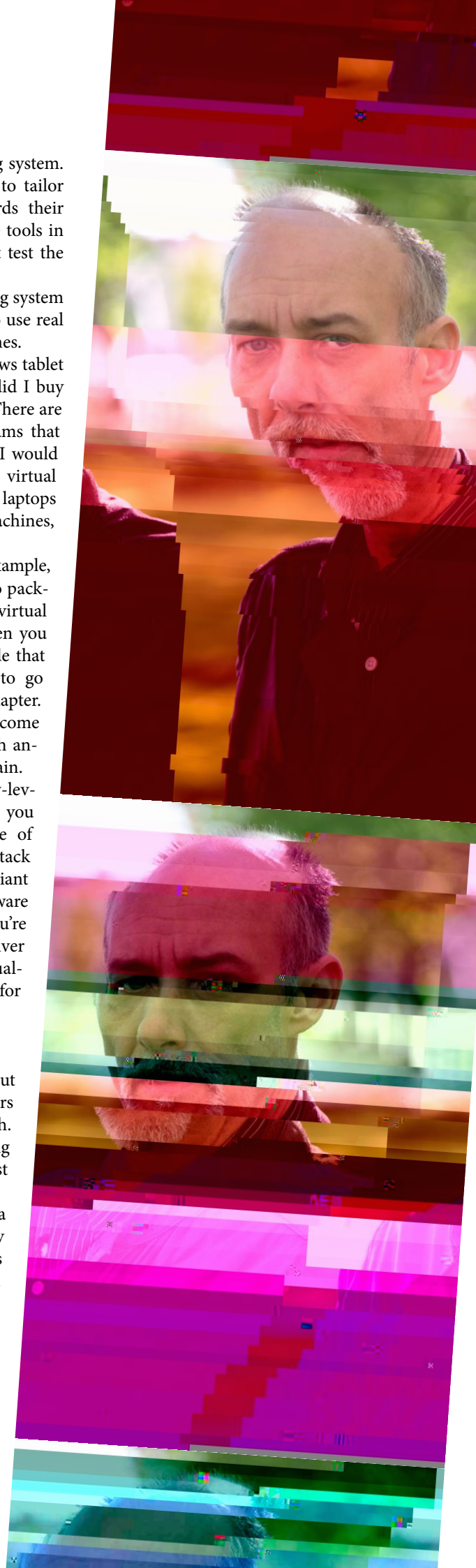
Demoralisation

Of course, we talk a bit about Linux, which many hackers have a special relationship with. Donnie himself started using Linux in 1993, before the first official Red Hat distribution.

– It was Slackware. I was at a discount book store, and they had some Linux book that was five centimetres thick. It had a command reference and all this stuff and I thought, this is cool, I like this!

– And then I started to see Linux everywhere, and Internet this and Internet that.

Donnie started going to Linux user group meetings, but he says they left a sour



taste in his mouth.

– People would focus on what they were really super into and they can do really, really well, and once they figured out that you can't do that, you don't know about it, then that's all they would talk about. It became really bad in this way, at least for where I was.

He said that he stopped using Linux and the Internet completely for a couple of years, just as the others were getting excited about it.

– I am not the kind of guy to go for *fads*, he says.

Currently, he says that he uses both Linux and Windows, but his main operating system is Windows.

– You want to get stuff done, use Windows. For utility work, I use Linux. When I'm actively testing something, I am generally within a Linux environment. My current work computer doesn't support some of the tools I use, but I have most of the tools on both platforms.

Curiosity

We wander off to some other topics and get back to him and his hacker peers. There are some things that bring them together. Donnie describes it as playing.

– It keeps us sharp and [we do it] because we're curious. We like knowing that we can break something that you got paid a lot of money to make.

Curiosity, playfulness and interest in computers seem to reoccur in his stories. Everything Donnie says also brings to mind a unique view on the world and a different way of looking at things. When we ask whether his way of thinking is driven by a hacker mentality, he only gives an indirect answer:

– I only have the use of one hand, so I have to think constantly about how I'm going to do something. How I'm going to walk to the door, carrying a bag of groceries and my keys, with another bag, holding my laptop under my arm. It makes me think constantly. Others do not need to do this. They simply use their hands. I have to think about how to use my hand always.

Donnie's hacking is not motivated by an ideal, even though for many grey hats like him, hackers who traverse the border between illegal and legal, hacking equals activism.

– There's this hacker conference in the US I went to. It *was* a good conference, but it was basically all hacktivism, so you have just tons and tons of wannabes who are looking up to just a few people actually in the industry.

He's not interested in "fighting the system"; to him, hacking is about something else entirely.

> ***"I only have the use of one hand, so I have to think constantly about how I'm going to do something."***

– I'm more technical, much like many other hackers I know.

He also feels that what cybercriminals do, for example, counts more as work than what he does. He doesn't believe the criminals master very complex techniques; they merely operate within the boundaries of what they have been taught. They are driven by money, not curiosity.

A question about his black hat past makes Donnie defensive.

– I consider myself a grey hat rather than a black hat. I can say I was a former black hat, just simply because I really just did whatever I wanted. But I have boundaries. I didn't go after NATO countries or universities of higher education. I was not malicious at all; even when dumping databases, I just didn't do anything with the data.

– I never considered myself to be a black hat, mainly because I didn't do anything for monetary gain.

Responsibility

We are happy with the stories we've got, but would still like to know if Donnie has something he would like to share with our readers. He thinks for a while, and we have a short break before pressing the red record button on our smartphone again.

– Security is about perspective, how you're looking at things. You should ask 'would I let my mother, the people that I care about, use this?' If you're ok with that, then you're probably doing okay. If you would question that, then do something about it. Take it upon yourself to learn more.

Again, Donnie gets lost in his thoughts for a while.

He recommends that you ask your employer if you can receive additional training for developing your skills. You need to know your weaknesses in order to improve.

He asks that developers take responsibility for their work performance.

– You may be doing okay, because developers are needed and you can do many things, but your employer has already spent a lot of money integrating you into their business. That's a big investment already. So they're more than likely quite willing to pay for you to go to a SANS course, or to take some extended course, especially if it's relevant. You can bring that back to your group as well.

– So go to them, and say 'I really enjoy what I'm doing. I really like this. However, I'm uncomfortable about some aspects.' The employers value you. They're going to pay for it. Take responsibility for yourself and expose your weakness.

To him, failure is nothing to be afraid of, and he emphasises that he himself fails 99% of the time. Also in situations where others succeed.

– I would've never found the current flaw that I spoke about earlier had it not been for that, for just trying something. I just shouldn't have even tried, but it worked, and it surprised me.

Pride

The sun is setting and the street lights in Ruoholahti turn on, one after the other. Donnie puts his empty beer glass on the table and we start to make our leave.

Our interview raised many different thoughts, and luckily, a digital record of our discussion is stored on the smartphone. Perhaps the foremost feeling is that Donnie is really proud of the work he does and does it with passion.

– Currently, my employer is the world's largest supplier of critical telecom infrastructure. The security of the world and its nations depend on our equipment. I am doing everything within my power to ensure their security, and I take this very seriously. Not that I believe so much in the company; instead, I believe in the industry that is providing critical infrastructure globally.

– This is how I apply myself.

The author wishes to thank Tapio Berschewsky, Laura Pesola and Anssi Kolehmainen for their help with this article.

Esoteric programming languages
Accordion solos as while loops

Programming is one of the most common tasks in the computer world, and anyone who has ever written code has most likely also become frustrated with the development environment and programming language at some point. However, what if the language was intentionally difficult to understand or it required the user to calculate base-three remainders? Esoteric programming languages are a large but backwards family that stretches the boundaries of programming.

Story by Jussi Kasurinen
Images by Sakari Leppä, Jussi Kasurinen, Wikimedia Commons: Rocky Acosta, Wattcle

Most people working with computers know a few programming languages – by name, at least – such as C, Java and Python. They also know that building the most interesting things requires hard work and dedication. It is also a generally accepted truth that programming is one of the hardest skills to learn, especially in the case of unusual disciplines, such as low-level system programming. What, then, is the purpose of programming languages that are designed to be substantially more difficult than normal ones?

More pain for little gain

Esoteric programming languages, or esolangs, are a family of programming languages where the basic concept of the language, such as the theme or design concept, takes priority over usability. In practice, this means that an esoteric programming language may be designed on the basis of a play script or a minimalist command library, such as in the case of **Brainfuck**, the world's best-known esoteric programming language.

```
+[>+<+++]>--.>++++++++[<++++>-]<
--------.+++++++.---.---..---.
```

Example: Brainfuck source code that outputs the text "Skrolli"; the structure of the entire language is based on a command set of 8 syntax characters.

Esoteric programming languages are rarely mentioned in the media or other literature, despite the fact that there are over three hundred such languages and many of them are Turing complete. So far, programming languages have been defined on the basis of cooking recipes, route instructions, songs and 19th century modern art, among other things. What these languages have in common is that, in theory, they could all be used to solve any logical problem or construct a completely normal graphical operating system.

Typically, esolangs are divided into five categories: minimalist languages that aim to be as compact as possible, thematic languages that are based around a topic, such as a deck of cards, conceptual languages that are based on a new concept, bizarre languages and joke languages that mainly have enter-

tainment value.

The most commonly featured esoteric languages are the thematic and minimalist variants, since they contain the most original ideas. For example, the **Whitespace** language that uses only non-printing characters, such as spaces, line breaks and indents, rose to fame through an April Fools' Day joke on the technology site Slashdot; similarly, **LOLCODE** was introduced to the public in an article on bizarre programming languages published by a newspaper in Houston.

```
HAI
CAN HAS STDIO?
VISIBLE "HAI WORLD!"
KTHXBYE
```

Example: Hello World in LOLCODE. The entire syntax of the language is based on Internet memes.

The most commonly known minimalist language is Brainfuck, which is also probably the only esoteric language that most programmers may have heard about. The language features Turing complete syntax built with eight single-character commands and a Turing complete compiler that was once the world's smallest; in compiled machine code form, it could fit in the space of 240 bytes, or two Twitter messages.

Undoubtedly, Brainfuck is one of the most impressive esolangs; it has over 120 different variants and a substantial amount of literature has been written regarding it. Overall, it could be stated that esoteric languages that are minimalistic, experiment with new concepts and technology or contain a specific theme are the most interesting and warrant further examination.

Technologically challenging pioneers

In many cases, the development of esoteric programming languages is not simply random fun or a pointless waste of time, since the structure of the language contains a generally useful feature or the need to demonstrate the functionality of an innovation. As a historic example of a language based on a thought experiment, we can mention **P"** that was developed by computer engineer **Corrado Böhm** in the mid-1960s. The language was based on Böhm's concept of creating the simplest possible command set that can replicate the Turing machine, and, thereby, creating a system for performing any operation on any computer, since the compatibility between computers was very unusual at that time.

Even though the language was marginalised, it is still one of the oldest platform-independent programming languages. P" was only a few years behind Fortran, the first portable language, and in theory it was a much simpler way to create platform-independent source code. Another language that has also been successful outside of the esoteric world is **Biota**, which is based on 2-dimensional source code that can be read in an arbitrary direction. Even though the language has not received widespread attention, partially due to the success of **Befunge** that applies a somewhat similar idea, Biota was a commercial success in its time, since the creator managed to commercialise the idea that started as an esoteric language experiment.

Out of the technologically challenging languages, we can also mention **Malbolge**, which is deservedly considered the most challenging programming language in the world and also an exercise in language interpretation that exceeds several old military encryption algorithms in terms of overall complexity. Malbolge is named after the lowest level of Hell in Dante's Inferno, and for a long time, it was considered practically impossible to use for the creation of any meaningful source code. It is fitting that the 99 Bottles of Beer example was published seven years after the announcement of the language, and that there are only a handful of known, working example sources available for Malbolge even today.

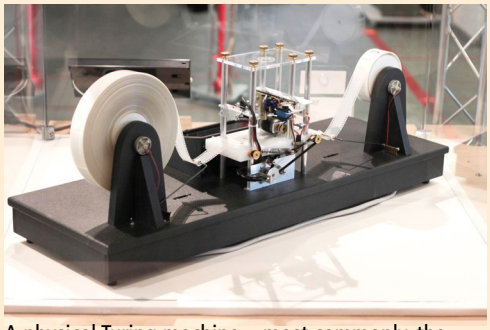
```
(=<`#9]~6ZY32Vx/4Rs+0No-&Jk)"Fh}|
Bcy?`=*z]Kw%oG4UUS0/@-ejc(:'8dc
```

Example: Malbolge source code that outputs "Hello World!"

In practice, coding is Malbolge is not so much difficult as it is laborious. First of all, the language uses ternary digits for everything; instead of the standard binary system, it employs ternary digits that include a value of two in addition to zero and one. The commands are provided as modifiers for two

Turing machine, Turing tar-pit and Turing completeness

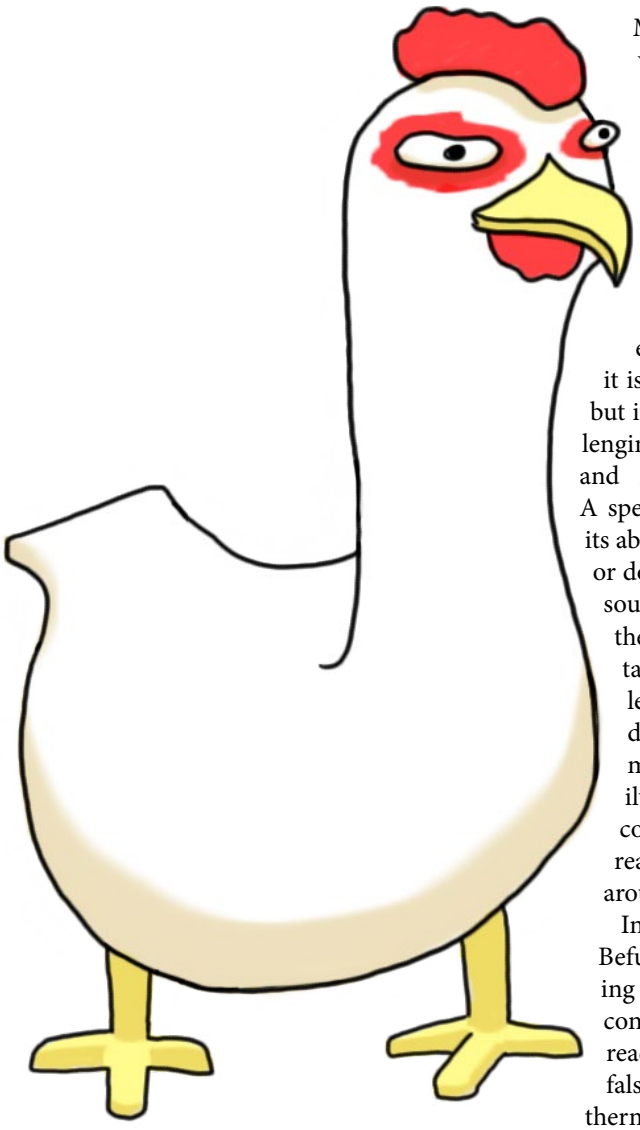
Turing completeness is the most common benchmark for the capability and functionality of a programming language. It means that the language can implement all the instructions in the Turing machine and includes all of its components. The Turing machine itself is a theoretical calculator developed by Alan Turing in the early 1900s that can read program code from a provided command table and react to input read from an indefinitely long memory tape.



A physical Turing machine – most commonly, the machine appears as a theoretical model.

In theory, all modern computers are based on the concepts of this machine, and it allows for solving any mathematical or logical problem that can be computed. For example, Brainfuck or P", the oldest known esoteric language, are more or less practical implementations of the Turing machine, and all real programming languages (Java, C, Python etc.) are also Turing complete, as are most developed esoteric languages. However, this is not self-evident, since Befunge and Malbolge were not Turing complete upon introduction, as both had severe limitations related to memory space.

Turing tar-pits, on the other hand, are often brought up in connection with esoteric languages. The definition comes from a phrase coined by Adam Perlis: "**Beware of the Turing tar-pit in which everything is possible but nothing of interest is easy.**" In practice, this refers to a situation in software development where the tools or methods being used prevent all effective and useful activity.



memory registers; for each command, the remainder of these is calculated in order to determine what the program does. The value table of the commands to be executed is altered after each executed command in order to prevent systematically reoccurring structures in the source code. Furthermore, the command set is designed in a manner where the manipulation of the values is performed by using trit operations on the trinary values stored in memory or with an instruction that converts trits according to an internal formula.

There are a few shortcuts to using Malbolge that are based on repeated number series, but in reality, the conversion algorithm used in the source code is substantially more complex than the ADFGVX message encryption algorithms used by Germany and the Austro-Hungarian Empire during World War I, for example. In fact, most of the features that simplify the use of Malbolge have been invented by means of crypto-analytical approaches, and the first working source code was created by a heuristic generator algorithm and not a human.

Befunge is another technically challenging esolang; unlike Malbolge, it is not difficult for the user, but its structure makes it challenging in terms of compiler and interpreter development. A special feature of Befunge is its ability to move right, left, up or down in a two-dimensional source code matrix. Since all the commands and the syntax being interpreted have a length of one character, the direction of code execution may be changed arbitrarily, in which case the next command to be executed is read from a cardinal point around the character.

In practice, this means that Befunge uses true code forking in its case structures; if the condition is true, the code is read to the right, and if it is false, it is read to the left. Furthermore, Befunge can modify its source code during execution. In theory, therefore, the language could use its own source code as a form of permanent memory or optimise its performance based on the execution-time results.

Both of these characteristics, the arbitrary direction of execution and execution-time modification, make the language very difficult for writing a comprehensive compiler that can create permanent machine code. Even though this has been successfully done, an easier practical solution is to create an interpreter as a combination of a simulator layer and a memory layer. With the exception of the unusual direction of execution and certain special characteristics, the language is almost remarkably normal. Memory management is based on a conventional First In First Out (FIFO) stack, the commands include all normal calculations, comparisons and logic operations, and all the text outside of the execution path is considered to be comments.

```
>25*"!dlrow ,olleH":v
                    v:,_@
                    >  ^
```

Example: Hello World in Befunge. The code inserts the characters in the stack and uses the arrows to form a physical loop that repeats until all the characters in the memory stack have been printed out.

New concepts as the structure of a language

In a sense, a computer is a fairly simple device. For example, source code does not exclusively need to be written in boring text format, even though this is arguably one of the easiest ways to read, write and edit program code. Although computers make it possible to surf the Web, send email and move around in three-dimensional game worlds, at the hardware level it is a straightforward device with memory addresses, registers and communication buses between them. When you drill down past all the embellishments and arrive at the lowest level, where hardware is controlled, everything works with a set of simple machine code instructions.

The lowest-level instructions are along the lines of "read a character from a memory address", "move to the next memory address" and other similar actions that happen behind the scenes. Since these instructions can be run on several different components, such as the processor cores or the GPU chip, at a rate of millions of operations per second, the end result is an illusion that the computer is capable of complex thinking, such as interpreting the gestures created by the user on the touchscreen.

Since programming languages work

Hello World in Piet.

```
Hello World Cake with Chocolate sauce.

This prints hello world, while being tastier than Hello
World Souffle. The main chef makes a ” world!” cake,
which he puts in the baking dish. When he gets the sous
chef to make the ”Hello” chocolate sauce, it gets put
into the baking dish and then the whole thing is printed
when he refrigerates the sauce. When actually cooking,
I’m interpreting the chocolate sauce baking dish to be
separate from the cake one and Liquify to mean either
melt or blend depending on context.

Ingredients.
33 g chocolate chips
100 g butter
54 ml double cream
2 pinches baking powder
114 g sugar
111 ml beaten eggs
119 g flour
32 g cocoa powder
0 g cake mixture

Cooking time: 25 minutes.
Pre-heat oven to 180 degrees Celsius.

Method.
Put chocolate chips into the mixing bowl.
Put butter into the mixing bowl.
Put sugar into the mixing bowl.
Put beaten eggs into the mixing bowl.
Put flour into the mixing bowl.
Put baking powder into the mixing bowl.
Put cocoa powder into the mixing bowl.
Stir the mixing bowl for 1 minute.
Combine double cream into the mixing bowl.
Stir the mixing bowl for 4 minutes.
Liquify the contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
bake the cake mixture.
Wait until baked.
Serve with chocolate sauce.

chocolate sauce.

Ingredients.
111 g sugar
108 ml hot water
108 ml heated double cream
101 g dark chocolate
72 g milk chocolate

Method.
Clean the mixing bowl.
Put sugar into the mixing bowl.
Put hot water into the mixing bowl.
Put heated double cream into the mixing bowl.
dissolve the sugar.
agitate the sugar until dissolved.
Liquify the dark chocolate.
Put dark chocolate into the mixing bowl.
Liquify the milk chocolate.
Put milk chocolate into the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Refrigerate for 1 hour.
```

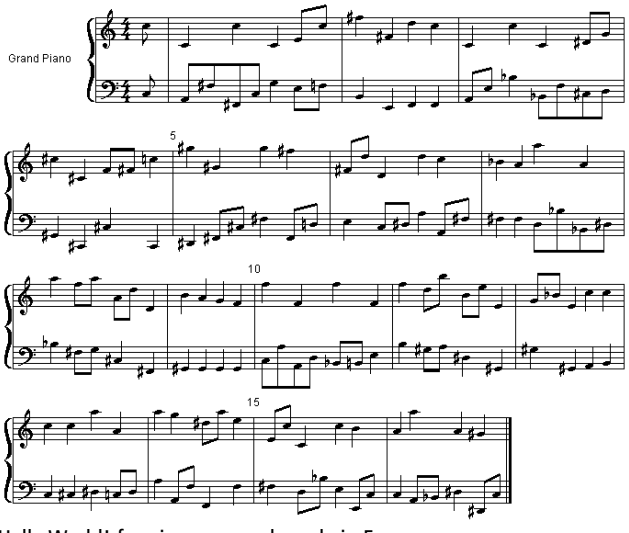
Example code 1: Chocolate cake recipe for Hello World in Chef. The specialty of this example code is that it is also a recipe for chocolate cake, although the sponge cake will be slightly dry. Other working source code recipes for Chef include a Hello World soufflé.

```
DO ,1 <- #13
PLEASE DO ,1 SUB #1 <- #238
DO ,1 SUB #2 <- #108
DO ,1 SUB #3 <- #112
DO ,1 SUB #4 <- #0
DO ,1 SUB #5 <- #64
DO ,1 SUB #6 <- #194
DO ,1 SUB #7 <- #48
PLEASE DO ,1 SUB #8 <- #22
DO ,1 SUB #9 <- #248
DO ,1 SUB #10 <- #168
DO ,1 SUB #11 <- #24
DO ,1 SUB #12 <- #16
DO ,1 SUB #13 <- #162
PLEASE READ OUT ,1
PLEASE GIVE UP
```

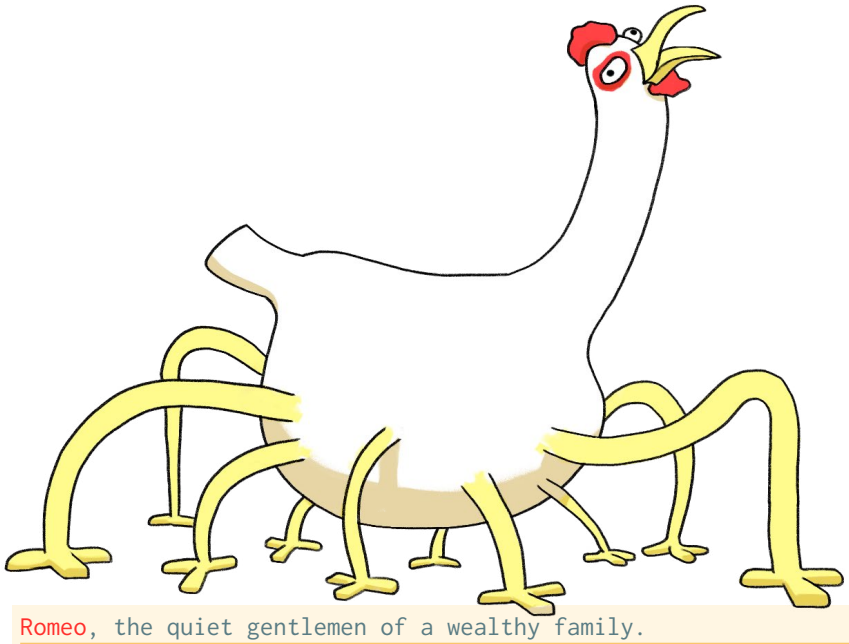
Hello World. INTERCAL was one of the first programming languages intended as a parody.

```
~ bottles {
  latlh 1 rap’a’ ”” tam
    { woD ”s” } ghobe’chugh
  ” bottle” tam tlheghrar tam woD
  ” of beer” tlheghrar
} pong

~ print-verse {
  latlh latlh bottles tlheghrar ” on the wall” tlheghrar
cha’
  latlh latlh bottles tlheghrar cha’
  ”Take one down and pass it around” cha’
  wa’boqHa’ latlh latlh  { ”No” } ghobe’chugh latlh
bottles tlheghrar ” on the wall” tlheghrar cha’
  ”” cha’
  latlh { print-verse } HIja’chugh
} pong

99 print-verse

~ morebottles {
“No bottles of beer on the wall,” cha’
“No bottles of beer.” cha’
“Go to the store, buy some more,” cha’
“99 bottles of beer on the wall.” cha’
} pong

morebottles
```

Example code 2: 99 bottles in Var’aq.



Hello World! for piano, example code in Fugue.



```
Romeo, the quiet gentlemen of a wealthy family.
Hamlet, the insulter of disrepute.


                    Act I: Hamlet's insults and exodus.

                    Scene I: The insulting of Romeo.

[Enter Hamlet and Romeo]

Hamlet:
 You lying stupid fatherless big smelly half-witted coward!
 You are as stupid as the difference between a handsome rich
brave hero and thyself! Speak your mind!

[Exeunt Hamlet and Romeo]
```

Example code 3: Snippet of Shakespeare source code. In the example code, variable Hamlet changes the value of variable Romeo by taunting him on stage.

in a manner where a compiler or interpreter creates a machine code version of the source code, this means that the source code created in a development environment is simply a set of instructions. The manner and format of entering the commands depends on the programming language used and is in fact trivial from the computer's perspective, as long as the machine code compiler understands which instruction it needs to use.

This theme is discussed by the "new concepts" subset of esoteric languages; in **Fugue**, commands are entered via MIDI compositions, while **Piet** uses images as its source code. There is also a language called **Light Pattern** that uses the metadata, colour values and histograms from photographs, and several projects that use smileys or emojis to generate code.

Fugue is a language that uses MIDI music as source code files. The language has ten different commands that allow it to perform all the necessary functions for Turing completeness. It also allows for using several sound channels, i.e. parallel or simultaneous sounds, to describe processes occurring at the same time. This means that the language supports parallel processing.

Fugue uses a specific formula to determine the command from the change in pitch compared to the previous note. For example, the first bars of the children's song "Gubben Noak", CCCE/DDDF/EEDDC, create the normalised source code

```
(first note) # # +  v # # + v # v # v
```

that removes values from the memory stack while moving to the next stack.

Fugue also has adaptations of Hello World and a few other basic programming exercises. The Hello World song sounds like a modern art performance and Quine (a program that prints its source code), with its repetitions, sounds like a long accordion solo in a sailor song when the default instruments are used.

The gift of speaking in tongues

Thematic languages form the final main group of languages that are mostly Turing complete. Typically, they include a concept that should be used for the syntax and layout of the code. Examples of this might include **Chef**, where the source code takes the form of a recipe, or **Shakespeare**, where the script of a play contains the code.

In Shakespeare, the actors are variables and can switch values by complimenting or taunting the other actors on stage. The interpreter recognises dozens of different taunts and compliments and uses their combinations to construct a binary value. Other lines such as "*Speak your mind*" are syntax commands that cause the other variable to print out its value as a character on the screen, for example. Similarly, in the Chef language, variables are provided as ingredients in a recipe where their initial values are provided as counts, and the stages of the recipe describe the operation of the program itself. Partial recipes, such as the frosting in the example, can be used to describe the internal functions of the program.

However, a more esoteric example of a thematic language is **Var'aq**, a programming language that is completely built on Klingon syntax. The syntax words are taken directly from the Klingon dictionary that linguist **Marc Okrand** compiled and systematised from the template created by the writers of the TV show. This seminal work has sold more than 300,000 copies, and during the peak of its popularity, the language was estimated to have more users than many actual minority languages. Klingon is also one of the few artificial languages to have its own ISO 639 code, and it also used to have its own Wikipedia variant, so a native programming language was the natural extension.

Technically, Var'aq is similar to Lisp,

and it includes about fifty different built-in functions or other operators and that makes it one of the most technically advanced esoteric languages and an actually working language. Var'aq also supports English syntax vocabulary, but you would have to be a batlhbe' programmer to resort to this – and, anyway, the language only comes into its own in the original Klingon language. Originally, the intention was to develop the language in three stages into a Klingon operating system, but development seems to have come to a halt after the first stage, the creation of a working programming language. There have been no major developments since 2004.

Esoteric benefits

Esolangs are not designed to be either simple or user-friendly. In most cases, the design and implementation of the language are at the level of "*yes, I can turn a bus schedule into a programming language*". However, complexity or obscurity to the uninitiated are not exclusively a feature of esoteric languages; very few can state off the top of their heads that the Perl statement

```
$string =~ s/^\s*(.*?)\s*$/$1/;
```

removes all the initial and final spaces from the input string. On the other hand, the lack of a serious purpose is not exclusive to esolangs, either; for example, the Logo language and its turtle were mainly created in order to have something that is easy to teach, and there was no intention to create actual software with it.

The problem with esoteric languages is that they lack a seminal work, although Brainfuck is arguably a fairly well-known language that no one has ever used. In this context, a seminal work could refer to a programming project that would create interest in the languages, much like Hamlet in Klingon. It showed that it is possible to create one of the best-known dramas in a language that was created from a television series and that lacks a verb for to be. Such a project could be an operating system in Brainfuck or perhaps a MOOC programming course in Befunge.

There is not a lot of information available on esoteric languages, but the Esolang Wiki (esolangs.org) is probably a good starting point for further research. The site collects esolangs and information about them. The Esolang Wiki also contains an idea section where you can find undeveloped concepts for programming languages in case you want to try building a language or a compiler. Egyptian hieroglyphs or the Voynich manuscript might be good ideas for the next esoteric language. All in all, the list contains over two hundred different concepts and ideas that have not yet been developed into full languages.

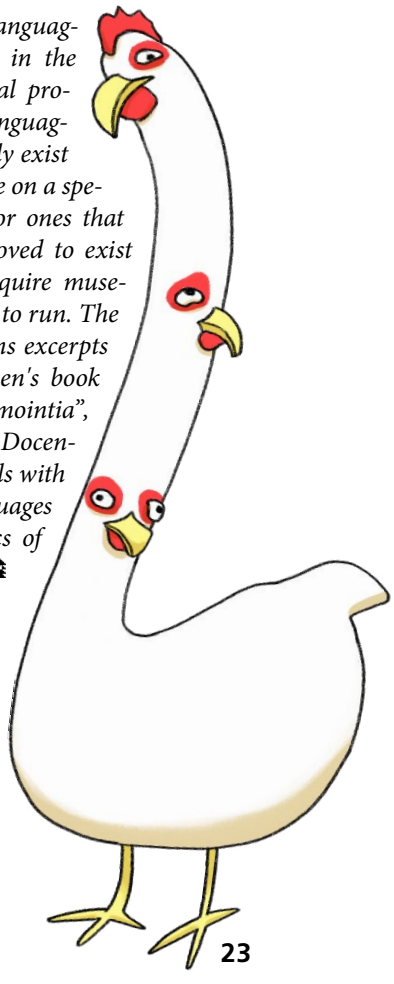
From an esotericism point of view, the most important feature is the elegance of the technical execution – or at least the entertainment value provided by the language. The fact of the matter is that Befunge is not likely to be used for browser applications, and Whitespace might not be the first option for a commercial software project. In the end, you could also say that an esoteric language with a useful purpose is no longer esoteric. It becomes a cryptic, normal programming language. But where's the fun in that?

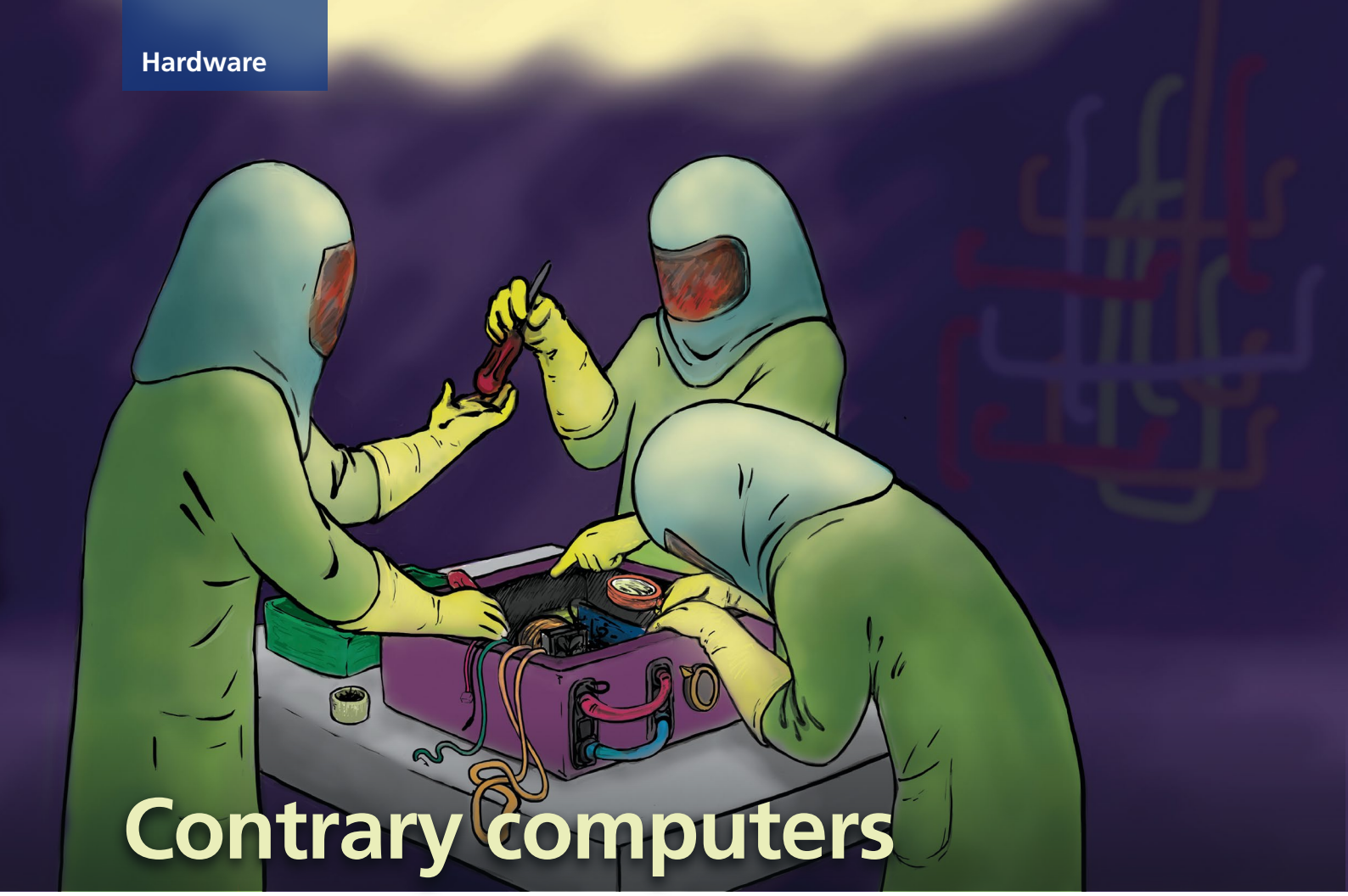
A timeline of esoteric languages

As demonstrated by the wiki site **Esolang** (esolangs.org), esoteric languages have existed for over 60 years. Despite this, most esoteric languages were developed at the turn of the 1990s, when the widespread popularisation of home computers started. Here is a short timeline of the most important events in the history of esoteric languages:

- 1930s: Alan Turing develops the Turing machine (1936), which is generally acknowledged as the first basic theory for the modern computer.
- 1940s: Conrad Zuse develops a theory of a platform-independent programming language known as Plankalkül (1943); if Zuse had followed through with his plan, he would have been over 10 years ahead of his time.
- 1950s: Platform-independent programming languages that allow for transferring software between computers are born (Fortran, 1957).
- 1960s: The research that presents the structure for P″ is published (1964). P″ and, later on, Brainfuck are practically real-world implementations of the Turing machine. The Logo programming language (1967) is published; its primary purpose is not to create software but to teach programming.
- 1970s: The INTERCAL language is published (1972). This is the first programming language that is specifically designed to be completely different from all other languages.
- 1980s: *Adam Perlis* publishes his 130 Epigrams on Programming; the concept of the **Turing tar-pit** is defined.
- 1990s: Biota, the first two-dimensional programming language is published (1991); FALSE and Brainfuck are published (1993). **Funge-98** (1998), the third version of Befunge, achieves Turing completeness. Malbolge, the world's most difficult programming language, is published in 1998.
- 2000s: Whitespace, a language that only uses non-printing characters, is published (2003). Piet, which uses images as source code, is published (2001). Velato, which uses MIDI music files as source code, is published (2003). ORK, the first object-oriented esoteric language, is published (2005). The first meaningful Malbolge examples are generated (99-bottles, Hello World!).

All of the languages mentioned in the article are real programming languages that actually exist and are usable on a specific system, or ones that have been proved to exist but would require museum hardware to run. The article contains excerpts from Kasurinen's book "Outoa ohjelmointia", published by Docendo, which deals with esoteric languages and the basics of computing.





Contrary computers

Let's assume that you want to build a computer that is completely different from anything designed before. Could you?

Story by Ville-Matias Heikkilä

Images by Mitol Meerna, Ville-Matias Heikkilä, Wikimedia Commons users Daderot, Shieldforyoureyes, Jitze Couperus, Rama, Rainer Lehrig, D-Wave Systems, Andrei Kulikov

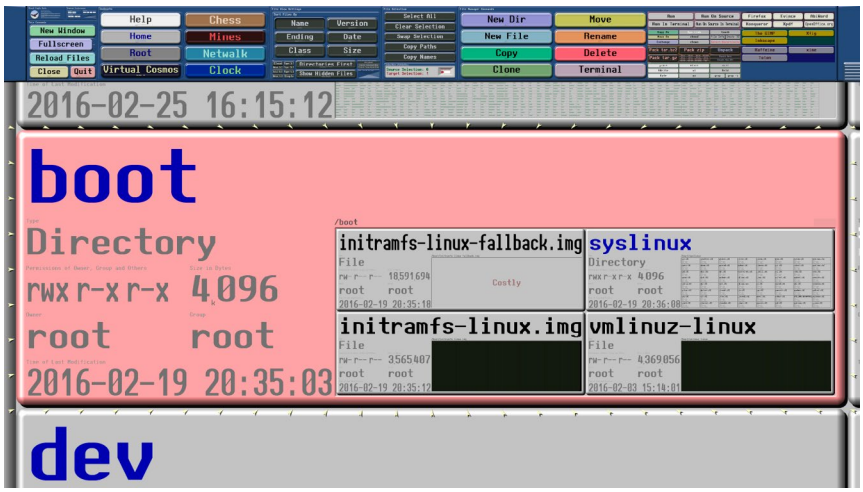
Find a friend who is not well versed in computers and ask them to imagine a system that aliens from outer space would use. Most likely, the concept will be based on the type of computer that they are most familiar with, such as a Windows PC, with some alien touches. This is usually the level that films and TV shows reach: a graphical user interface may have odd shapes instead of familiar letters and triangles in place of squares, but no one questions the very idea of a graphical user interface.

An expert who has experience with many different computers from different eras can provide a much more educated guess, but even they may get stuck in the obvious, unquestioned assumptions. The world of computers is full of established technological solutions that are by no means the only ones possible.

External appearance

It is easy to imagine computers that look different. There are no theoretical limits to what a computer can look like, and there are plenty of examples of this in our everyday lives. Embedded systems are found everywhere, from small everyday objects to large building complexes. As soon as nanotechnology develops further, computers do not even need to be solid; every gas cloud may be a computer.

However, if the intention is to use the machine effectively, we need to set some limits to the appearance of the machine or the terminal attached to it. Most people primarily observe the world with their eyes and interact with



Zooming user interfaces are often considered bizarre. The Eagle Mode file manager includes one, for example.

their hands, so computers have traditionally come with display screens, blinking lights, switch panels, paper printers, VR headsets and pointing devices. It is fairly easy to imagine viable alternatives for these, even if we stick with the eyes and hands. Even the nanoparticles in a gas cloud computer could produce visible patterns and observe the user's movements.

When studying existing user interfaces below the level of input and output, most of them may be classified as tangible or linguistic. In the tangible paradigm, the user manipulates the objects inside the computer in a more or less direct fashion. It may, however, employ some sort of movable cursor or another type of virtual avatar. Linguistic user interfaces, on the other hand, are based on command interpreters and question prompts from the computer. However, there are some intermediate forms, such as text editors in the vi family and Sierra On-Line's old adventure games.

A system where the computer uses a neural network to learn from the user's example might fall outside of this dichotomy. This seems to be the direction where basic user software is headed anyway, as they are already trying to guess what the user wants to do. However, something like the Pure Data development environment would also fall outside of this division, as its structures are based on the graphical interconnection of blocks instead of language.

Historically, future computers have been envisioned to contain the extreme form of the current user interface philosophy. During the reign of the command line, futurologists were certain that man and machine would interact in natural language – even spoken language. Nowadays, we might dream of user interfaces where you have even more opportunities for shaping objects with your hands or ones where the machine is even better at guessing what the user is trying to do. Therefore, in order to ensure that your user interface is completely unlike anything else, note down these trends.

From surface to core

A regular user's understanding of the inner workings of a computer is based on the operation of the user interface. At times, the surface describes the insides fairly well, while at other times, the user interface is a facade that protects the user from the operating system. For example, smartphones often obscure the entire file system and directory hierarchy from their users, even though their Unix-based operating systems are very often built around the concept of directory trees.

You can, therefore, create an illusion of an entirely new type of computer by creating a user interface that fools the user in a different way than the earlier ones. But what if cheating is not an option? In this case, the operating system and user interface must operate within the same conceptual realm. This is the case in the original Macintoshes, for example.

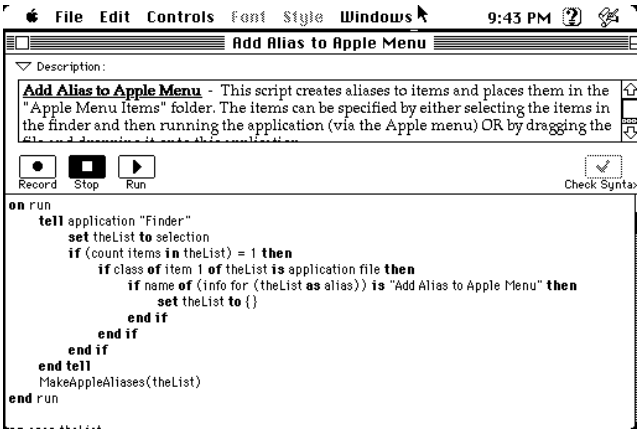
In the classic Mac OS file manager, there is a direct correspondence between the icons and the files on the drive. The file system contains dedicated fields for the graphical location of the file as well as the file type, which is not encoded in the file name in any way. The structure of the file is also exceptional. While Unix-type files may be presented as byte sequences, the files on a classic Mac have two forks: the data fork contains the basic data, whereas the resource fork is used to save additional items such as icons.

The designers of the Macintosh were eager to question the status quo. To them, a terminal-based command line was a relic of outdated thinking, and it was not available even for software developers. The command interpreter in Macintosh Programmer's Workshop (MPW) is used from a worksheet; it can be freely edited like a text file, but pressing Enter will run the command at the cursor location.

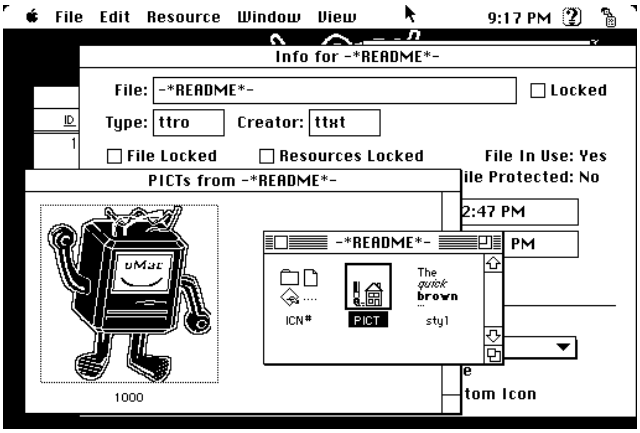
The language used by MPW is reminiscent of Unix command interpreters. However, the aim was to make end-user script languages, such as Hypertalk and AppleScript, as similar to natural language as possible. At one point, the goal was that AppleScript could be read and written in the user interface language, even in international variants.

The differences of the Mac world had several effects. Mac users could easily understand their machine and its errors, but interoperability with other brands hit a veritable wall of incompatibility and incomprehensibility. And different problems arose when the interfaces of other computers tried to imitate the Macintosh: The Amiga Workbench will not recognise a file unless it is accompanied by an .info file that includes the icon and other Mac-type metadata. Windows has tried to hide file extensions from the users, which has caused confusion.

Nowadays, even Macintosh itself



Apple wanted to make writing – or even recording – scripts as easy as possible, even for basic users.



Dissecting the internal structure of a Mac text file. The text is in the data fork, while the resource fork even includes graphics.

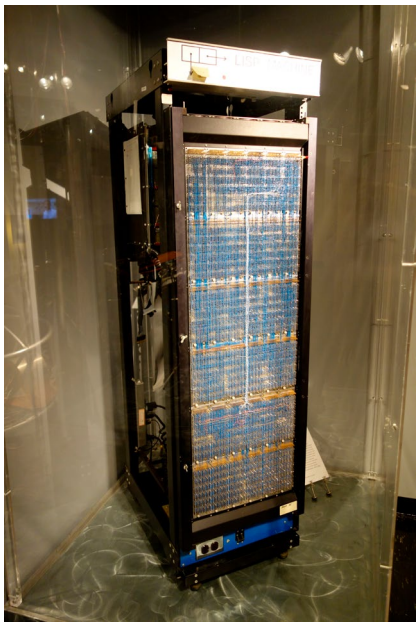
is guilty of cheating. The Unix based OS X requires a facade similar to Windows in order to appear like a Macintosh. A genuinely different approach that reached the very core of the operating system was replaced with "Think different" campaigns, and even the long avoided text terminal appeared in the Applications menu.

The forgotten AI machines

The original Macintosh may be considered an ideological machine wherein each area is designed to support ease of use and a graphical user interface. However, it is by no means the only ideological machine. Sometimes, the high-level design philosophy has been extended all the way to the machine code level.

In the early 1970s, artificial intelligence researchers at MIT were frustrated with the fact that their language of choice, Lisp, was not a good fit for the computers of the era. The decision was made to design a workstation from the ground up from a Lisp perspective. Its processor, instruction set and operating system would be completely built on Lisp and its philosophy.

If a computer based on a high-level programming language sounds odd, it should be borne in mind that mainstream computers are also designed for specific languages and ways of thinking. If Lisp machines had been mainstream from the beginning, the computers we now know might be referred to as "C machines" or "Fortran machines".



CADR was the first serially manufactured Lisp machine.

The main difference between the operation of Lisp machines and mainstream computers is the use of typified data. Each binary word in memory includes a few bits that signify whether the data is an individual number or a list pointer and where the next list element might be found. A dedicated chipset performs type checking in parallel with the actual computation; even in compiler-based Lisp implementations for Fortran machines, this took up most of the computing power.

Nowadays, Lisp machines are a relatively obscure side note in the history of computing, but they were firmly believed to have a future in the 1980s. Lisp was considered to be a strong and expressive language for future AI applications as well as a host of other things. The machines were commercially manufactured by the company Symbolics, and their Genera operating system borrowed many pioneering ideas from the Xerox Alto graphical workstation. As I write this, Symbolics has been bankrupt for 20 years and their machines have largely been omitted from the history books. The Emacs text editor is perhaps the most visible legacy of the Lisp machines in the modern world.

However, Lisp machines were fairly down to earth and ordinary compared to the "Fifth Generation Computer Systems" (FGCS) project that Japan's Ministry of International Trade and Industry started in 1982. The project was intended to revolutionise the world of computing and completely renew many of its foundations. Computation would have consisted of AI-driven knowledge processing, based on massively parallelised logic programming

and Prolog as the "native language". During the project, however, Prolog was replaced by a logic programming language known as KL1.

In the end, ten years of development created five different Parallel Inference Machines (PIMs) that could perform at most a few hundred million logical deductions per second. Thinking Machines, a contemporary American company, had already given up its AI focus much earlier, and its Connection Machine supercomputers were mostly used for ordinary computing.

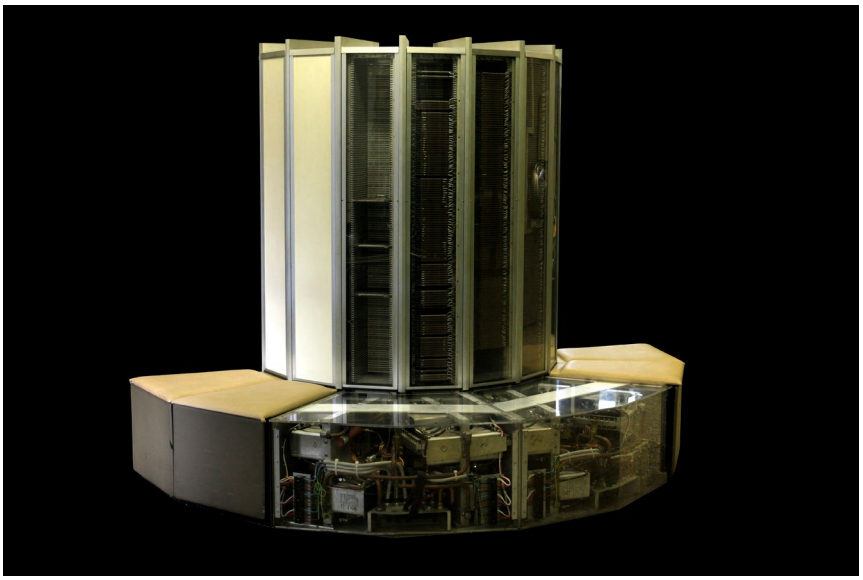
Cray's number crunching monsters

When discussing computers whose every feature demonstrates a specific philosophy and vision, we must not forget **Seymour Cray**. Cray designed supercomputers from the 1960s until the 1990s. Each time, he set out to design a machine that would outperform any previous design. Anything that did not support this goal was questioned. Very often, Cray's team also achieved its goals.

Cray's first supercomputer was the

The Space Cadet keyboard from a Symbolics Lisp machine: a mixture of familiar and unfamiliar keys.

CDC-600 supercomputer from the 1960s, with control desk.


Cray-1 from 1976. Instead of aesthetics, the eccentric design is based on the optimisation of component distances and heat transfer.

CDC-6600, completed in 1965. In a time where most computers were running software in clear and consecutive steps, the 60-bit CPU on the 6600 distributed simple instructions to several parallel calculation units. Other machines favoured a CISC approach, with single expressive instructions, but the CDC-6600 had a RISC instruction set – some twenty years before the term was even invented. The CFC-cooled system could perform some three million floating point calculations per second.

Some principles were repeated in Cray's designs, decade after decade. One of them was that number-crunching hardware must not be made to wait for mass storage, user interaction or input and output devices. The CPU had to be employed at all times, so separate auxiliary processors and front-end machines were used for trivial tasks and supplying data to the CPU.

Even during the golden age of blinking light panels, Cray refused to install one in his machines, since operating one would have required stopping the CPU. Instead, the console of the CDC-6600 had two round vector displays that showed a control interface operated by auxiliary processors. The machine must have been a monstrosity to contemporary operators, as they could not control the electronic brain directly and had to interact with a bug-eyed mediator machine.

Overall, Cray did not take kindly to solutions that would have made the user's life easier at the expense of performance. Virtual memory is a good example of this: it is impossible for the machine to work effectively if the programmer cannot and does not have to decide which part of the data is in the main memory and which part is on a mass storage device. Eventually, Cray consented to using microprocessors with the introduction of the Alpha in the 1990s.

Multitudes of computing

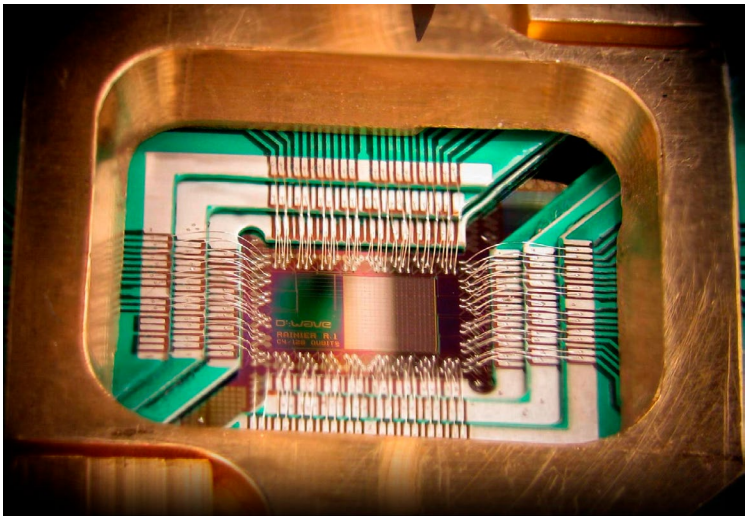
A key feature in Cray's machines in the 1960s and 1970s was the limited parallelisation of single-thread consecutive code – which is also used in most modern processors. However, development could have taken another turn.

In the 1980s, INMOS in the UK was developing Transputer CPUs that were designed to connect to other Transputers via high-speed links. If Transputers had been a success, the number of processor cores might already have been a selling point for consumer hardware some 10–20 years ago, and a parallel aspect would have been introduced into basic software development much earlier. However, Transputers were mostly used in embedded systems, and the only notable generic Transputer system was Atari's failed Transputer Workstation. The ATW could use up to 13 20-MHz T800 Transputers.

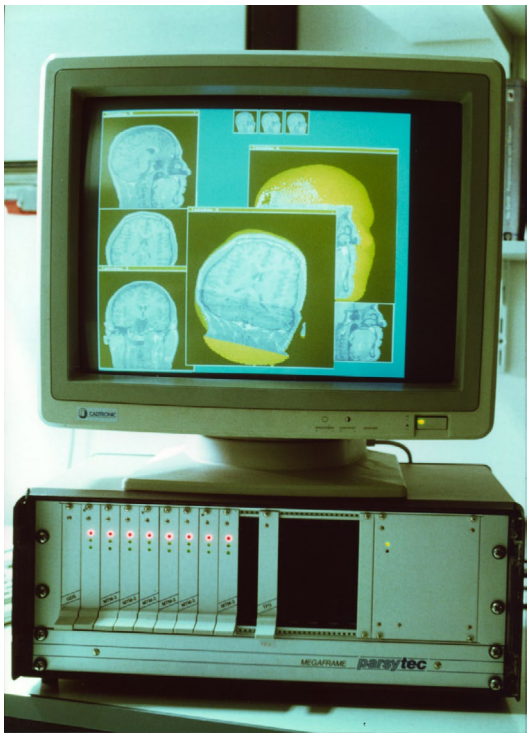
Nowadays, the Transputer dream has become true: ordinary computers have multiple parallel cores and graphics processing units that are especially well suited for parallel calculation. And, surely, this is only the beginning. Going forward, computers are likely to contain even more eccentric calculation units. These include, for example, programmable logic chips (FPGA and rDPA), neurocomputing units and quantum processors.

Programmable logic may be seen as microchips with reprogrammable contents. So far, these chips have been mainly used to replace Application Specific Integrated Circuits (ASICs) that are expensive to manufacture in small volumes.

However, programmable logic has even more applications. It allows for reconfigurable computing, where the hardware can be modified "on the fly" to suit the task at hand. Rainer Harten-



A 128-qubit quantum processor from D-Wave.



The Megaframe workstation from the German Parsytec could fit a maximum of ten Transputer processors.

stein's Xputer is one possible model for reconfigurable computing. Hartenstein has also referred to reconfigurable computers as "anti-machines" with reference to the fact that their "hardware" is not static.

Neurocomputing units are intended for use as platforms for non-biological neural networks. The strength of neural networks is that they do not need to be programmed in the traditional sense – the weight and bias network that forms their logic is created automatically. This makes them particularly suited for computer vision that is difficult to break down into modules that a human programmer can understand. In a few decades, even human minds may be copied from the brain and be run in neurocomputing circuits.

Quantum computing is based on the idea that a suitably constructed computer can exist at the quantum superposition, i.e. have multiple states at the same time. Put simply, the machine may be considered to compute different things in different parallel universes: if, say, 8 of the machine's quantum bits are in the superposition, the computing is parallelised into 256 realities. Quantum computing is especially suited for tasks where you need to find the correct solution among a vast number of possible answers. D-Wave from Canada introduced the first working quantum computer in December 2015.

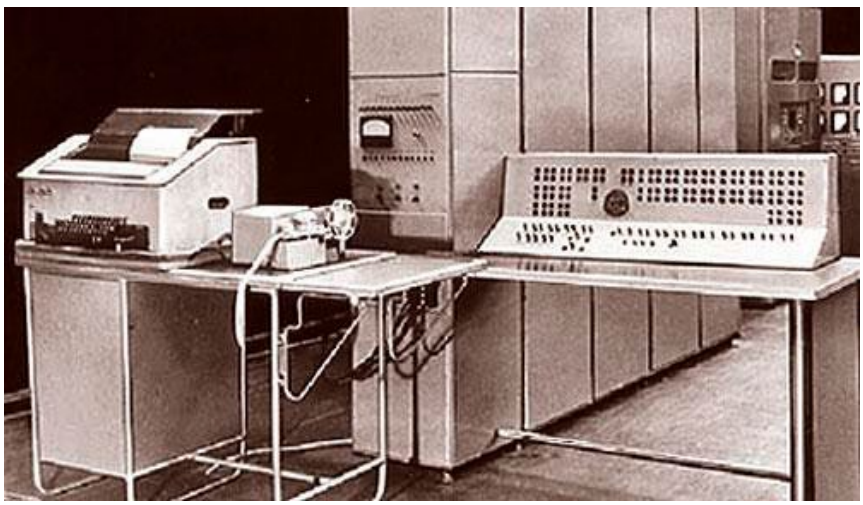
Beyond ones and zeroes

Transputers were possibly overshadowed by RISCs and "riscified" CISCs because they required such major changes in thinking and programming. In mainstream computing, new paradigms are mostly accepted when they can be subordinated to the traditional way of thinking. Even if the machine is full of complex computational logic, it still needs a familiar consecutive processor and a traditional operating system to control it. The GPU has an important role in a modern PC, but a machine controlled by the GPU would be considered heresy.

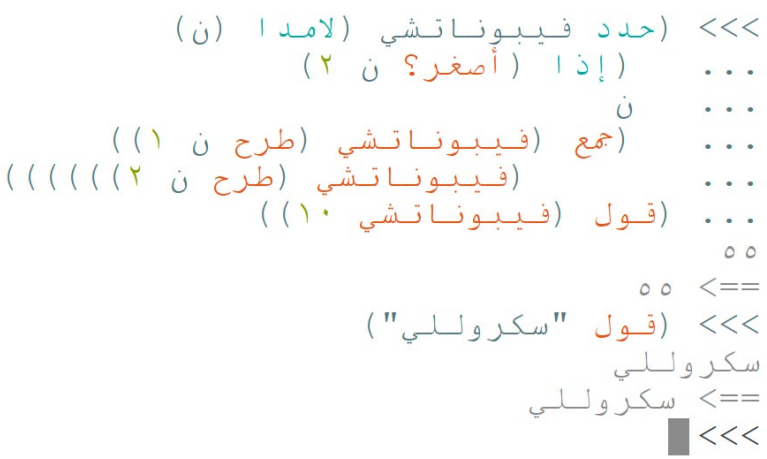
When addressing a versatile group, the language tends to follow the ideals of the dominant culture. GPU-based parallel computing languages such as GLSL and OpenCL are very close to traditional C. Therefore, we should not assume that the hypercomputing of the future would lead us to question such obvious basic assumptions as the binary system, for example.

Already in the 1950s, computers were mostly using three number systems: binary integers, binary floating point numbers and binary coded decimals. The University of Moscow, however, was convinced of the benefits of a balanced trinary system, and the idea was tested in the Setun computer completed in 1958 as well as its successor, the Setun-70.

Whereas a binary system is based on ones and zeroes, the balanced trinary system includes a third option: minus one. One of the benefits is that negative numbers never need to be pro-



Control desk and other hardware from the Setun trinary computer.



Qalb is an Arabic Lisp variant. Its developer **Ramsey Nasser** wanted to study the effects of the writing system on the programming experience, and has even created calligraphy works from some programs.

cessed differently from positive numbers, and rounding and multiplication circuits are also less complex. Setun was a very elegant and simple design and it attracted attention from capitalist countries as well. However, the Setuns became the only trinary computers ever built, even though the idea does reappear in research articles once every few years.

Traditional Aristotelian logic, which is also the basis for the Boolean algebra used in digital technology, is binary in nature: a statement is either true or false. Therefore, binary logic does not work very well with unknowns, conflicts and both – and conditions. If trinary machines and their programming techniques had become more commonplace, software might be more tolerant of exceptions.

Technology has its roots in culture

The triumph of binary logic may be seen to express the internal binary nature of our culture: "You are either with us or against us." European languages are much better at expressing either–or conditions than more complex relationships, so it was natural for us to develop Aristotelian logic, Boolean algebra and binary computers. If, however, computers were developed by the Aymara people in South America, for example, it might be trinary in nature, as their language offers better building blocks for three-valued logic.

Language alone may have had a fundamental effect on the development of computing. However, this is hard for us to observe, since English has been such a dominant language. Most programming languages – even ones that use keywords in another language – are based on the structure of English. Programming languages have no declension and the relationships between words are described through strict word order.

However, there is at least one programming language based on Finnish; an obscure educational language from the 1980s called Sampo that uses case endings as keywords. If a programming language was constructed exclusively based on Finnish, it might primarily express structural relationships through declension. However, the use of declension within programming languages is currently such an obscure idea that it has not even been widely considered by the esoteric language community.

You can also analyse the effects of different cultural features on computing by deconstructing them in a post-modernist fashion. Hierarchical data structures may well be a reflection of the hierarchical thinking that permeates our culture, and the emphasis on black boxes may be a symptom of the alienating nature of our industrial society. Even if the justifications seem far-fetched, this kind of thinking allows you to find features to question within the field of technology and thinking, and to look for interesting alternatives.

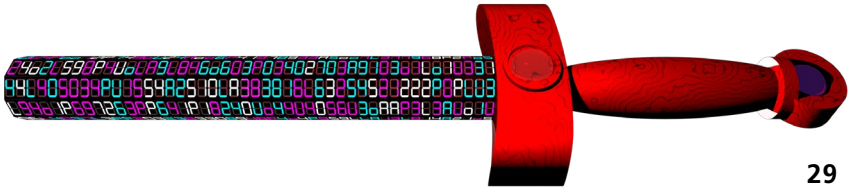
A couple of years ago, computer scientist **Ari Schlesinger** set forth the idea of a feminist programming language that would be completely built from a feminist starting point – starting from the programming paradigms and logic systems. Traditional binary logic would be replaced by quantum physics-inspired paraconsistent logic, and a less normative manner of abstraction would be used in place of object-oriented programming. We are still looking forward to more concrete further developments of this idea.

Building your vision

It should be possible to design some fairly eccentric computers on the basis of the ideas in this article. For example, how about a computer that resembles a temple and that you control by performing rituals inside it? Or what about a machine whose logical foundations, computing paradigms, programming languages and software are based on cyclical thinking instead of the traditional, linear Western concepts? Ideas such as these are fun to play around with, and they can also broaden your views even if they are not generally technologically feasible.

Fundamentally different technological concepts may require decades of work to mature. Large groups of researchers worked on Japan's AI machines for ten years. Furthermore, despite two decades of research, reconfigurable computing is still not generally available. So if you really want to start something new, you should have enough faith in your vision to spend countless hours on it.

Over the decades, computers have become increasingly similar. Despite the superficial differences, there are only a few different processor architectures and operating systems, all of which offer similar philosophies: productised black boxes, deep hierarchies of industrial modules and preventing the user from understanding technology. Therefore, there will surely be demand for a completely contrary computer – so start designing!



All aboard the code train!

Computer control for model railways

Model railways are a versatile hobby for all ages. Nowadays, larger trains are computer controlled – like in real life.

Story and images by Mikko Rasa

I have owned a model railway for nearly my entire life. Over the years, my interest in trains has increased and expanded, and I can only try to guess what my original motivation might have been. The technology of the trains and remote control were probably significant contributing factors.

When I was little, I used to play with *Brio's* wooden trains. They were honest toys and built to withstand use by small children. In the 1980s, there were no battery operated engines, and you had to push the train along the tracks manually. You could also roll it down a hill, but it would cause it to derail quite often. My parents still have the Brio track and it is taken out when the children of my relatives visit.

Lego bricks were another childhood favourite, and I got a Lego train set in the early 1990s. The trains were electric and powered by the rails. I expanded the track on several occasions. At its longest, I had over 20 metres of track, including three or four sets of switches. In addition to the three freight wagons in the starter set, I received three more as gifts or bought them off my allowance.

Later on, I found a motor unit for an older train in a bag of Legos I bought from a flea market. It was powered by a third rail and was not directly compatible with my own rails. In the spirit of DIY, I built pickup shoes out of metal plate that allowed me to use the motor with the newer rails. I also built a simple signal that had a switch for changing the colour of the light. The same switch also controlled the passage of electricity to the subsequent track section, so that when the light was red, the train would stop at the signal.

Shortly after the Lego train, I also acquired an old Finnish *Märklin* catalogue from 1982. Compared to the standard-length straights, single-radius curves and oddly-shaped switches of the Lego train track, Märklin seemed to offer nearly endless possibilities. They even had remote controlled switches. However, a real model railway was so expensive that I could not afford it.

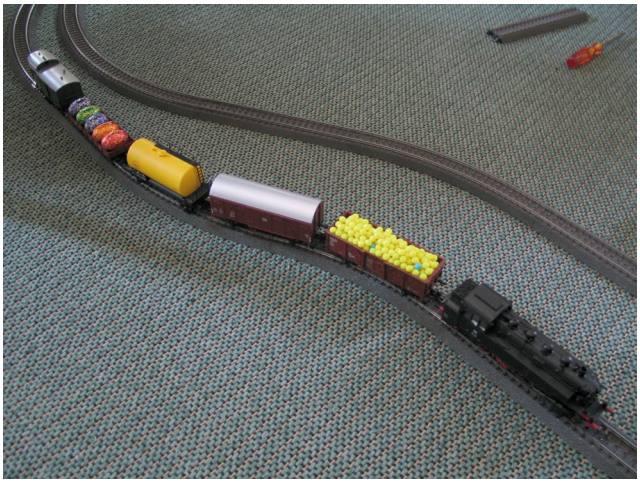
As the 1990s rolled on, the limitations of Lego tracks started to really get to me. At the same time, I was becoming more and more fascinated with computers. Then, in 1998, I made a major decision: I would sell the Lego track and use the money to buy a computer. This was the start of a dark age in my model railway hobby: it seemed that all the available options were either too toy-like or too expensive.

This changed when I found a permanent job in the autumn of 2005. As I now had the financial means, and the Internet offered a wealth of information, I started finding out what had been happening on the model railway front since the late 1980s. It turned out that trains were controlled by digital technology and you could also connect the track to your computer. The embers of my railroad hobby lit up once again. I had to have one of those! In the spring of 2006, I visited Model Expo to look at model railways and came back with a Märklin starter kit and an expansion set with more tracks. The track has expanded since, and I now own over 100 metres of track, some 25 engines and about two hundred wagons.

Before moving on to the technology behind computer-controlled tracks, I would like to give you an overall picture of the model railroad hobby and its different aspects.

Model railways in general

There are many sizes of model railways. The most common scale, which my set is also in, is H0 (1:87). Scale N, which is almost half the size (1:160), comes second. Other common options include scales 1 (1:32) and Z (1:220). You can proportionally fit more track in the same space with a smaller scale, but you lose details and functions. Scale Z is so small that no commercial



Here is where it started: the starter kit train on the living room floor.

In early 2016, my collection included 27 engines, over 200 wagons and over 100 metres of track.

digital control is available. On the other hand, even real steam engines are available for scale 1.

Nearly all scale H0 and smaller tracks are assembled on a table of some sort. The platforms vary from ordinary furniture to massive, bespoke fixed tables. The size depends on the scale and the builder's level of ambition. The smallest Z-scale tracks can fit inside a large suitcase, whereas the world's largest H0 model railway has a surface area of over 1,000 m^2.

Some hobbyists and, in particular, model railway clubs build their tracks out of modules. A module track usually consists of one-metre sections with ends that follow commonly agreed measurements and electrical connections. You can combine the modules freely to build larger systems, which allows hobbyists to come together and build tracks that none of them could do alone.

You can even build a model railway in your garden. Scale 1 or larger is recommended, since the larger size will make the trains less susceptible to uneven terrain. Large-scale rails are often designed to be weatherproof with outdoor tracks in mind. Nevertheless, you should keep your rolling stock safe from rain.

The tracks on a model railway serve many purposes. Trains run along them and receive their voltage and digital commands through them. Tracks fall in two main categories: ones with two and three rails. In a two-rail system, the voltage is connected to the rails and the trains pick it up through their wheels. The construction is simple, but reversal loops must be isolated in order to prevent short circuits. The axles of the trains must also be insulated.

In three-rail tracks, the running rails are connected to the same potential and they have an extra third rail between them. The engine has a pickup shoe that connects with the third rail. Since crossing the running rails does not matter, the track geometry may be freely chosen. The redundancy may also be utilised to build rail circuits that are used as sensors for the passage control system. The downside is the aesthetic drawback caused by the third rail.

Most model railways have a landscape environment. The landscape has visually appealing elements that attract visitors to the track during exhibitions. It may be based on an actual location or the builder's imagination. Exact copies of famous stations or other landmarks are common. Landscapes usually cannot be built to exact scale, since even in a small scale the distance between the stations may be 10 metres or more.

Real railroads include many other devices in addition to tracks and trains. Therefore, signals, overhead wires, lamp posts and other accessories are also available for model railroads. With the exception of the cheapest models, these are not merely static props; they also work in a similar manner to their larger-scale counterparts.

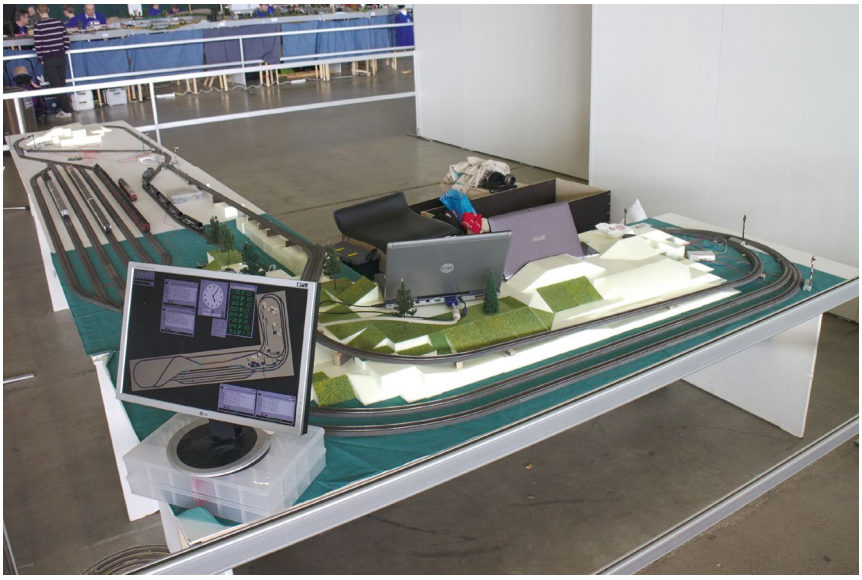
From analogue remote control to the digital age

In order to achieve an authentic atmosphere, it must be possible to control the trains without touching them. Simple analogue control is the oldest control system. The voltage from the track is connected directly to the electric motor on the engine, and speed is adjusted by changing the voltage. Multiple trains can be operated independently only if the track is split into several circuits. All trains inside the same circuit will run at the same time, but not necessarily at the same speed, since different engines may have different gear ratios.

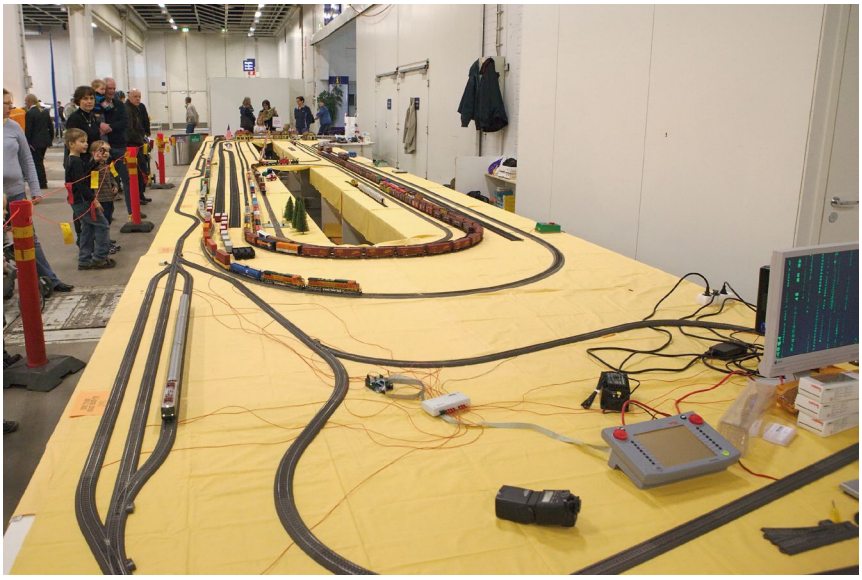
An analogue model railway controller is typically a box with a rotating knob for speed control. Depending on the system, you can either drive forward and backward by turning the knob above or below zero, or the scale has a separate point for changing direction. Large tracks require a separate controller for each control circuit.

The first digital control systems entered the market in the 1980s. Widespread adoption took some time, however, especially since old analogue engines cannot be used on digital tracks without modification. Digital control moves the speed control logic to the engines, which makes their speeds independent of each other. In the 2000s, manufacturers have started to discontinue analogue engines and controllers in all scales except for the very smallest ones.

Initially, digital control units looked similar to their analogue counterparts, but before long, manufacturers started adding additional function buttons and display panels. Basic controllers usually have one or two integrated control



The model railway from Model Expo 2015 as a digital and physical version.



Even larger layouts can be transported to events, provided that you do not try to build landscapes around them.

consoles with speed adjustment knobs. The device can send commands to more engines, but alternating the manual control is difficult. More advanced systems have a separate central unit that allows for connecting several different controllers for engines and railway yards alike.

In principle, a digital model railway is a simple data network that adheres to a bus architecture. Command packets consisting of ones and zeros can be sent over the bus by altering the polarity of the operating voltage. In addition to engines, the switches, signals and other accessories can also be digitally controlled. Each device connected to the track has a digital decoder and a unique address that allows commands to be routed correctly.

Since the data is modulated in the polarity of the operating voltage, only the controller supplying the voltage can send out commands. The first protocols were purely unidirectional. Some newer protocols also allow for sending data towards the controller, but only as a response to a request command. The return signal is formed by modulating the decoder's power consumption.

Unshielded cables, several connections between track sections, inductive loads and moving engines are likely to cause interference in the signal. In order to improve reliability, the data transfer speeds are fairly low. Depending on the protocol, the commands move at 5–20 kilobits per second. However, most control commands are only about 20 bits long. The protocol also adds a set of pulses for synchronisation.

The most widespread control protocol is known as *DCC* (*Digital Command Control*). It is an open standard that most model railway manufacturers adhere to. The basic version of DCC has 7-bit addresses and 5-bit speed control, but the standard also defines 14-bit addresses and 7-bit speed control as an option. DCC equipment is available from several manufacturers, some of which only build control electronics instead of rolling stock or rails.

Märklin Digital is the most important competitor for DCC. It consists of two different protocols. The older of them bears the Motorola name, since the first decoders were made by Motorola. In order to avoid confusion, it is commonly called *Märklin-Motorola* or by the abbreviation *MM*. In 2004, Märklin launched a new protocol known as *MFX*, but Motorola is still used for controlling accessories and cheaper engines. MFX has a 14-bit address space and 7-bit speed control.

MFX has two-way communication that allows the controller to read the information on the engine and its functions from the decoder's memory. The protocol also includes a handshake for identifying the engine placed on the track. After a while, the engine will appear on the controller's screen, and the user does not need to worry about remembering addresses or avoiding address conflicts.

Of the other protocols, at least *Selectrix* is worth mentioning. It has no separate commands; instead, the controller sends the contents of all 112 memory addresses in the system onto the track about 13 times per second. One memory address may contain the information for one engine, including a 5-bit speed setting, or the positions for eight switches or signals. Due to its synchronous nature, Selectrix offers unparalleled speed, but its features are fairly limited compared to later protocols. In addition, the fixed structure leaves no room for expansion.

Lately, it has become possible to install Wi-Fi receivers in the engines and control them over an IP network. This IoT solution allows for fully two-way communication and is not susceptible to interference caused by the rails. On the other hand, Wi-Fi frequencies may

This compact industry diorama combines exact landscapes with versatile trains and functionalities.

The details of this Alpine scene from Switzerland show the possibilities and visual appeal of detailed landscaping.

be quite congested during fairs and exhibitions. There are no ready-made commercial solutions yet.

The different protocols are not officially compatible, but since the structure of their command packets is different, they can be used simultaneously on the same track. Packets with incorrect frames or otherwise invalid packets are disregarded. MFX even takes into account the similarity of a specific bit sequence with DCC's initial synchronisation; if an MFX packet has eight consecutive zero bits they are followed by an additional one bit.

Not simply running around

So what do hobbyists do with their model railways? Building landscapes was already mentioned. A skilfully crafted miniature landscape is pleasing to the eye of the builder and visitors. The most dedicated hobbyists build their engines and wagons themselves in order to have them exactly right.

Operating the trains becomes more interesting when the traffic has a purpose. Instead of the trains simply running around in circles, they transport passengers and cargo from station to station. Some model railway clubs, such as FREMO-FIN in Finland, follow real railway practices with engine drivers, train dispatchers and radio telephones. Automation can also be used to assist with the simultaneous operation of multiple trains.

Model railways can also be a collector hobby. There are many beautiful scale models of engines and wagons available, and the manufacturers keep renewing their ranges. Dozens of different versions of the most popular engines have been released over the years. Collecting may also include building. A Finnish hobbyist has constructed miniatures of all the trams in Turku, for example.

A modern model railway has plenty of digital technology that you can study and utilise. There are both hardware and software based solutions available for controlling your track. The most active hackers develop their own accessories.

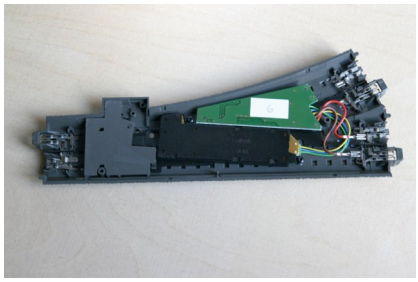
Computer controlling your railway

My first controller was the Mobile Station that came with the Märklin starter kit. It had no computer connection and it was otherwise limited as well. After the initial fascination of a real model railway had worn off, I returned to the store to acquire more supplies. The Central Station from Märklin was too expensive, so I chose the Intellibox by Uhlenbrock.

The Intellibox supports Märklin-Motorola, DCC and Selectrix, but not MFX. The device is a typical, integrated model railway controller: it includes speed adjustment knobs for two engines, buttons for controlling actions and a small screen for status information. In addition to the model railway connections, the Intellibox includes an RS-232 connection for the computer. It offers both a text-based protocol and a binary protocol, both of which have fairly comprehensive documentation.

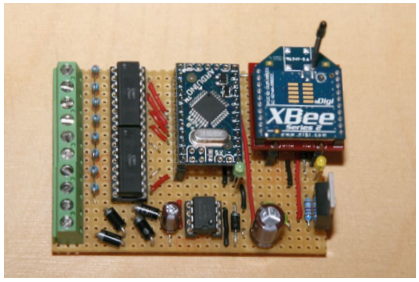
Over time, more and more engines with MFX support ended up in my collection. Although MFX decoders can also understand Motorola, I started to miss more precise speed control and an end to the address bingo. I was able to source a second-hand Central Station for an attractive price at a model railway event.

Central Station is a much more versatile device than the Intellibox, and it offers a graphical user interface and touch screen in addition to the speed knobs and function buttons. The first version, which I bought, only supports Märklin's own protocols, but this did not matter to me since I only owned

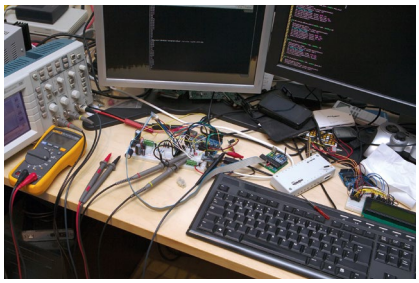


Control mechanism hidden under a switch.

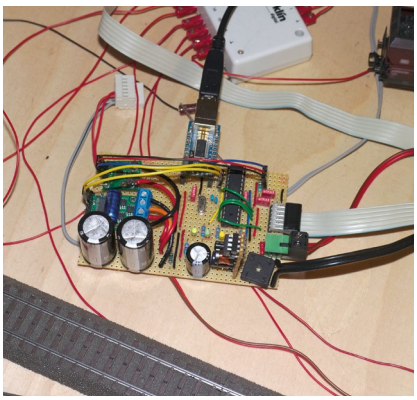
A model railway controller, such as the Intellibox in the picture, allows for controlling a model railway both locally and with a computer.



The first accessory I made: a wireless transmitter for track circuits.



The first version of my own model railway controller was quite a mess of wires.



The second version was cleaner and more advanced.

Märklin engines. Central Station connects to the outside world over Ethernet. I found the description for the remote operation protocol after some digging. It was partially lacking and completely in German, but made enough sense for me to be able to build a driver for my software.

Unfortunately, my Central Station failed after a couple of years and it could no longer control MFX engines. Repairing it would have been almost as expensive as buying a new one. I still was not ready to pay that much, so I temporarily went back to the Intellibox. At the same time, however, I started designing my own controller in order to regain MFX support.

The structure of a controller is fairly simple: a microcontroller and an H bridge. The latter are often sold as motor controllers, but they are also a perfect fit for the needs of a digital model railway. In addition, you also need the parts related to the power supply and computer interface.

Reverse engineering the control protocols and implementing them in a manner that Märklin's decoders accepted was much harder to do. I had already purchased an oscilloscope earlier, and it turned out to be indispensable for this work. Finding information on the older, simpler MM was easier. The protocol has somewhat bizarre encoding, but after a few days of tinkering I was able to get the engine moving.

MFX, on the other hand, required more detective work in order to find and understand the information. The only website I found was in German, but machine translation turned it into comprehensible English. The next thing to do was to visualise how the protocol works. In MFX, decoders have a fixed 32-bit ID, but shorter, dynamically distributed addresses are used for sending out the commands. The controller must perform a successful handshake with the decoder and provide it with an address – if this fails, commands cannot be sent.

Eventually, I was able to get MFX working. The largest challenge was to identify the 52.6 kHz carrier wave of the return channel that is modulated into the current draw. The motor controller capacitor I used caused the signal to disappear. I was able to solve the problem by moving the identifier chip to the track side. I am working on a new model with a different motor controller. The parts for the controller are less than €100 in total, which means substantial savings compared to the Central Station's price tag of over €600. Unlike other controllers, this one is not designed for standalone use, so it has no screen or buttons. The higher level protocol logic is also implemented on a PC.

Connecting your computer

In order to make your computer more than an oversized model railway controller, you need to be able to transfer the information on the track shape and train movements to it. The track diagram is built with design software that I will explain in more detail below. Depending on the rail type, there are a few different solutions available for tracking trains. My track uses Märklin's three-rail system that is particularly well suited for track circuits. In this case, you isolate one of the running rails from the rest of the circuit, and as the train enters a signal section, its full-metal axles will reconnect the isolated rail to the rest of the circuit.

I am also working on an infrared gate that is based on the train interrupting the infrared beam. Compared to a track circuit, this detector is very small. Infrared gates are especially suited for rail yards with limited space. On the other hand, their more complex structure makes them more expensive and laborious to construct.

Of course, the signals generated by the detectors must be transmitted to the computer. For this, I have designed and constructed devices that gather signals from 8–12 detectors and forward them via wireless link. For legacy reasons, the receiver is connected to the controller via S88 bus As the track size and the number of detectors increase, it may be necessary to connect the receiver directly to the computer's USB port in order to reduce latency.

I have also written the software I use for designing and controlling the track. The design software can be used to create a track diagram without the risk of running out of track. The exact dimensions of all the parts have been entered into the software, so I can ensure the geometric suitability of the track before constructing it. I can

also display the available tables, which will help in the design. Once the track is complete, I can use a separate utility to create a shopping list that takes into account the track parts purchased earlier.

Traffic control requires data

The really interesting stuff goes on in the control program. The program is ready to use once the track diagram has been loaded and the positions of the trains have been input. Sliders can be used to set the speed of the trains, and switches can be turned by clicking on them. However, this is only the beginning.

Passage control in order to avoid crashes is one of the most important and oldest functions in the software. Similarly to real railways, the system works based on absolute block signaling. The track is divided into blocks that may only have one train at a time. In order to move into a block, the train needs to reserve it first. If the block is already reserved, the train needs to wait for it to be released. The system creates the blocks automatically on the basis of the track diagram and shows the reservations on screen with colour coding.

Passage control requires up to date location data. The detectors can only provide binary data on whether there is a train at a specific point or not. However, the initial status and block reservations allow the software to determine which train it is. The data from the detectors is also used for measuring the speed of the trains and determining their more exact location. However, this extrapolated location may lose its synchronisation with the real situation, so the system uses it for passage control in a pessimistic manner.

The option to define routes in advance was also added to the program at an early stage. Usually, they were laps around the track. A train sets the switches in a position that corresponds to its route, so several trains can take their own routes even when sharing a track section.

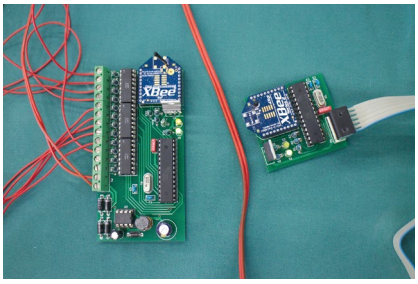
Manually, you can only run a limited number of trains at a time – even with passage control. Then again, trains that run on standard routes are fairly boring. Over time, I wanted more complex traffic on my track, so I created a simple script language for automated traffic. At the early stages, each train was completely independent of the others, but I also added synchronisation commands later on. In principle, this was a case of multi-thread programming, and it also brought with it the typical problems of parallel execution.

Since creating manual synchronisation for the trains was very susceptible to mistakes, I had to create a better alternative. In the current version, automatic traffic is based on timetables. The times use the 24-hour clock, but you can speed up the passage of time so that a day can last 30 minutes, for example. The program does not yet support timetables spanning several days.

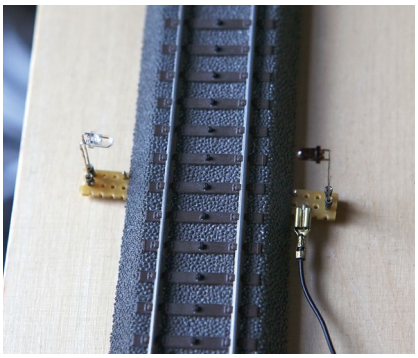
A route finder algorithm implements the timetable. It takes into account the schedules of all the trains and is capable of avoiding conflicts. The basis of the algorithm is A* with a state-space that consists of all possible combinations for the train locations. The heuristics are based on a distance map from every point on the track to the destination of each train, generated with the traditional Dijkstra's algorithm. As the execution proceeds, the algorithm keeps track of the rails reserved for each train and ensures that no two trains use the same rails at the same time.

The generated route plan contains sequence points where a train has to wait for another train. This ensures that a random malfunction will not cause unexpected routing problems, since delays are easier to compensate for than misplaced trains.

Being able to test the functionality of the track and timetables without a physical track is useful. To this end, the program contains a simulation mode. It internally generates the inputs that are normally received from hardware. It can also simulate delays and errors. The simulation can be sped up in order to test longer timetables.



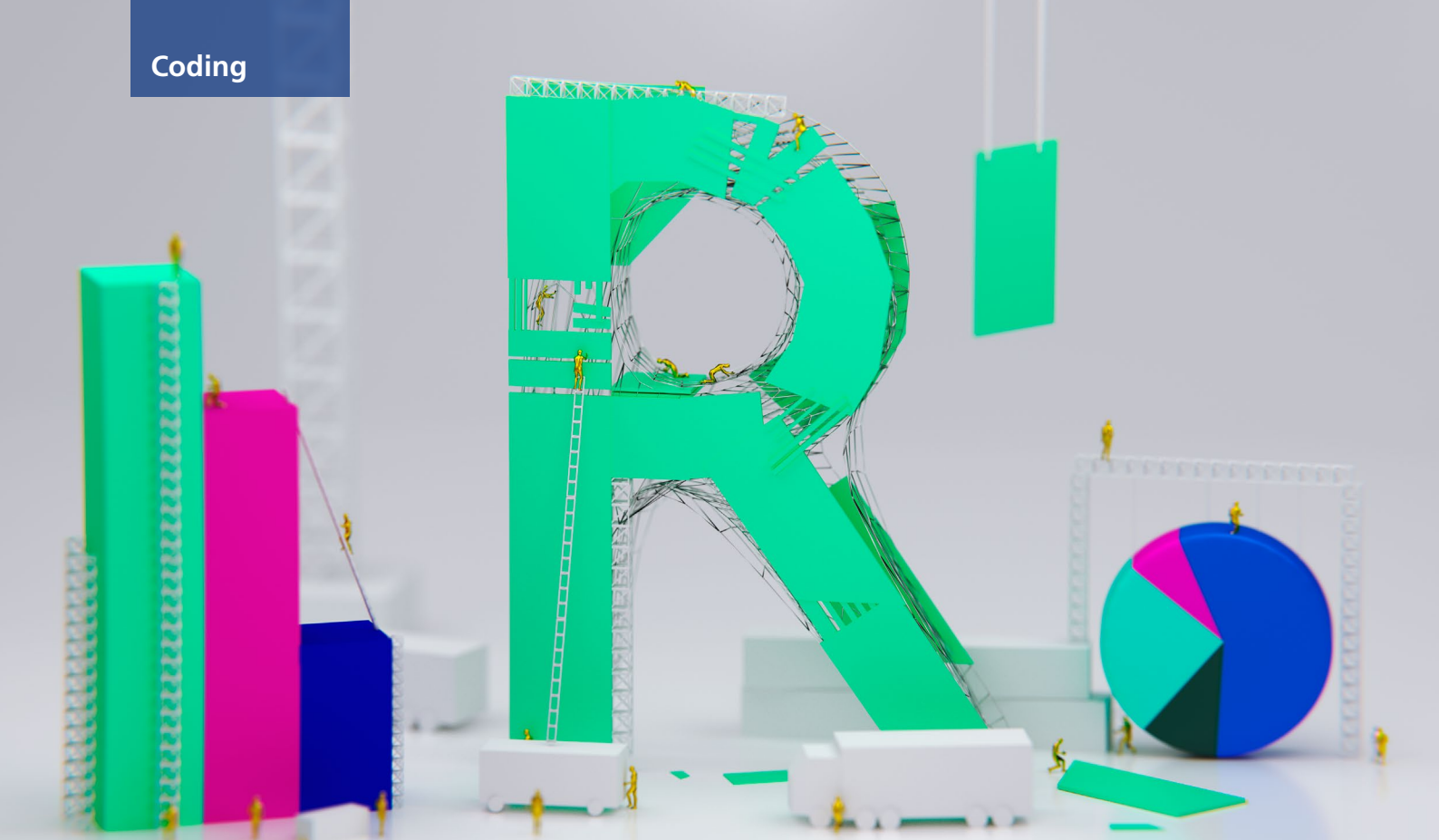
Transmitter (large) and receiver (small) for the wireless detector system.


An infrared gate.

This is one of those projects that will most likely never be completed. Future plans include better physics simulation and improvements to the user interface and graphics. In the long term, I want to add game elements into the software – imagine *Transport Tycoon* with a real model railway!

Web links on the subject

- www.tdb.fi/railway.shtml
  My model railway site
- git.tdb.fi/?p=r2c2.git
  Source code for my model railway software
- www.miniatur-wunderland.com
  The world's largest model railway
- www.fremo-net.eu/en/home/
  Site for the European FREMO module standard
- www.dccwiki.com
  Information on the DCC protocol



R – A language for data analysis

The open R language, designed for scientific calculation, challenges the commercial MATLAB language that has been dominant for a long time.

Story by Jarno Niklas Alanko, Jussi Paananen Images by Toni Kortelahti

MATLAB, developed by MathWorks in the 1970s, was the de facto standard language for scientific calculation for a long time. It is an interpreted language that requires a costly commercial interpreter. MathWorks has been selling individual licences at €2,000, taking advantage of its monopoly in the field. An important piece of the puzzle has been to offer discounted licences to universities and students in order to get people hooked during their studies. Later, companies have been charged ten times as much, since their employees already know the language and there were very few alternatives.

There is a free MATLAB interpreter known as GNU Octave, but its functionality has some limitations, and minor differences mean that code written for the MATLAB interpreter will often need to be corrected before it can run on Octave. For these reasons, Octave is mostly popular with students who want to do their MATLAB exercises at home without paying for the licence.

The community for scientific and statistical calculation does not need a watered-down MATLAB clone; instead, it needs a language that does what MATLAB does and more. R is such a language.

It is an interpreted language, which means that programs are not compiled into machine code as in C. Instead, they are input to a command interpreter that runs them. This allows for using the language interactively. You can work by opening the interpreter, loading in data and manipulating it interactively. It is possible to save the session and resume your work from the same point at a later time. Writing and running ready-made script files is also possible. For additional performance, there are ways to program demanding functions in C++, for example, and load them into the R interpreter.

Specialties within the syntax

The syntax of the language is based on the fairly obscure S language. It is much further from hardware than C, for example. There is no primitive data type for integers; instead, the basic data type is a vector and an individual integer is represented as an integer vector with a length of one. The standard library contains the most common mathematical operations for vectors, such as addition, scalar product and vector product. There is also a selection of functions for processing arrays and the basic tools for statistical analysis.

Versatile and flexible vector indexing is one of the handier features in the syntax. From the surface, the indexing looks familiar: if X is a vector, then `X[3]` produces the third element of the vector. A vector may also accept a list of indices. The command `X[c(1,5,6)]`returns a vector that includes the first, fifth and sixth element of vector `X` (function `c` combines its parameters into a vector).

The vector may also be indexed with truth values. The command `X[c(TRUE,FALSE,FALSE,TRUE, FALSE,FALSE)]` returns the first and fourth element of the vector. The command `X[X > 0]`, on the other hand, returns a vector that contains all the positive elements in the list `X`. The expression `X > 0` inside the square brackets compares each element of vector `X` with zero and returns a list of truth values that indicates for each

element of vector X whether the comparison with zero returned TRUE or FALSE.

Beginning users of R are often confused by the substitution operator <- (gets). For example, the command x <- 5 means that variable x is substituted with a value of 5. Later on, R was updated to use the equal sign as a substitution operator, so x = 5 will also work. However, there are some subtle differences in the functionality of the two operators, so beginners should always use <- to be safe.

R ecosystem

The community of scientific calculation, machine learning and statistics is already actively using R. New algorithms and methods are mostly first published as R packages. One of the key benefits of the language is that the latest algorithms and methods from the academic world are usually instantly available through the R interface. Packages are compiled in the Comprehensive R Archive Network (CRAN) system for anyone to download. Some R packages are written in languages other than R, but they can nevertheless be called directly from the R code over a standardised API.

The maintainers of the CRAN archive have fairly strict criteria for the quality and documentation of the code accepted into the archive. While this is a good thing, since it guarantees the quality of the packages, it also makes the approval process long and laborious. In part due to this, other archives have also been set up, such as Bioconductor which specialises in bioinformatics. Nowadays, many people are also using packages directly from GitHub with the develtool package developed by **Hadley Wickham**.

Wickham has also developed a large number of other very popular packages, such as ggplot2 for plotting graphs, tidyr for tidying data and dplyr for data management. Wickham alone has done more work for R in recent years than anyone else. Once you get used to Wickham's libraries, you never go back to standard – the difference is like night and day. Wickham has also written several free books on the use of R, its optimisation, package development and the use of the libraries he has developed.

```
sum_loop <- function(v) {
  s <- 0
  for(i in 1:length(v)) {
    s <- s + v[i]
  }
  return(s)
}

v <- as.numeric(1:100000000) # Add numbers 1..10^8 into a vector
system.time(sum(v)) # Internal sum() function
system.time(sum_loop(v)) # Self-made sum loop
```

Listing 1.

Avoid loops

The language is weak in classical algorithmics. The standard libraries lack many essential data structures for algorithmic programming. For example, there is no dynamic list that you could add elements to in constant time, not to mention search tree structures. On the other hand, they do offer a data frame structure that is useful for representing statistical data; it is a two-dimensional table whose columns and rows may be named and indexed with the names.

Traditional loop structures are slow, but operating with vectors and ready-made functions is swift. For example, it is much faster to calculate the sum of a vector with the existing sum function than to loop through the vector individually with a for loop.

You can demonstrate the slow speed of the for loop by creating a vector that contains numbers from one to one hundred million:

Save the code as sum_benchmark.R and run it in the R interpreter with source("sum_benchmark.R"). It turns out that the built-in sum function takes 73 milliseconds, but the for loop version takes 20 seconds; 273 times more. Effective use of the language requires a programming style where vector operations are used to perform heavy calculation. The basic functions in the standard library have been designed in accordance with this style; they can be called with a vector, and the function effectively operates each element of the vector.

From data to knowledge

One feature where R is clearly superior to its competitor MATLAB is the plotting of graphs. Hadley Wickham's excellent ggplot2 library is available for plotting. The library is based on the *Grammar of Graphics*. In normal use, however, knowledge of grammar theory is not a requirement; unless you are creating a very innovative graph, using the built-in assistant functions is sufficient.

However, the fact that ggplot2 is based on the Grammar of Graphics allows for constructing a unified, logical interface for plotting different diagrams. Unlike in MATLAB, for example, all the charts consist of similar elements and that makes it easy to plot many types of graphs and effortlessly switch from one graph to another.

The grammar divides all the information required for plotting a graph into six individual parts: data, geometry, statistical conversions, scaling, coordinate system and facets. These components precisely define what is plotted in the graph. In the Grammar of Graphics, different graphs, such as line charts, bar charts and pie charts, are merely special cases of a more generic graph concept. The system can create these classic graph types, but you can also combine different types; if you know the grammar extensively enough, creating completely new graph types is also possible.

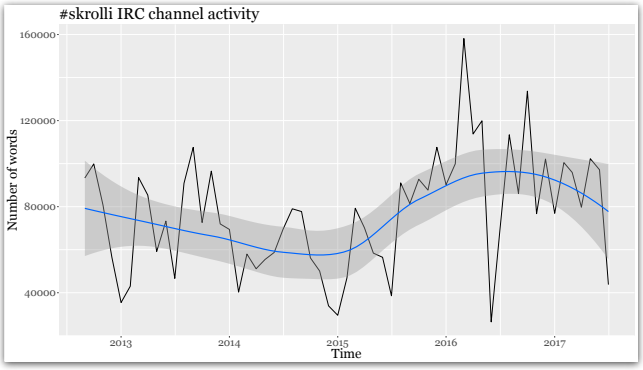


Figure 2. The black line indicates the number of words per month and the blue line is the local regression curve for the black line. The grey area shows the 95% confidence interval for the model.

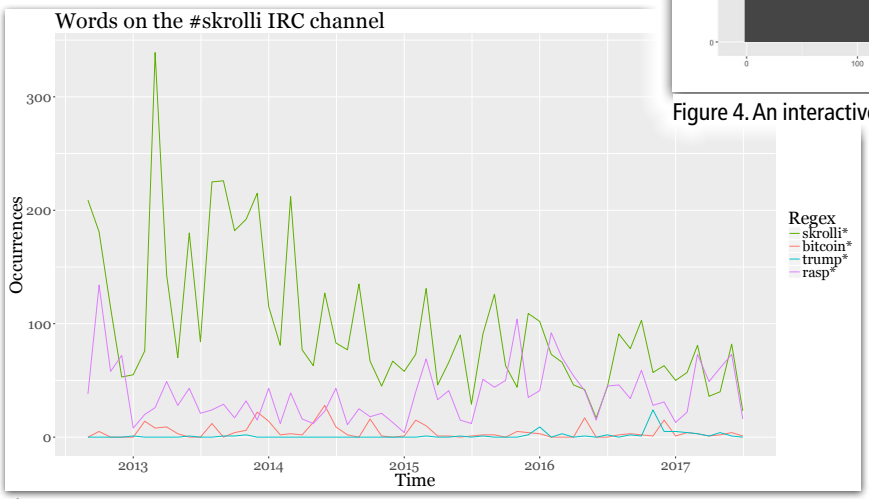


Figure 1.

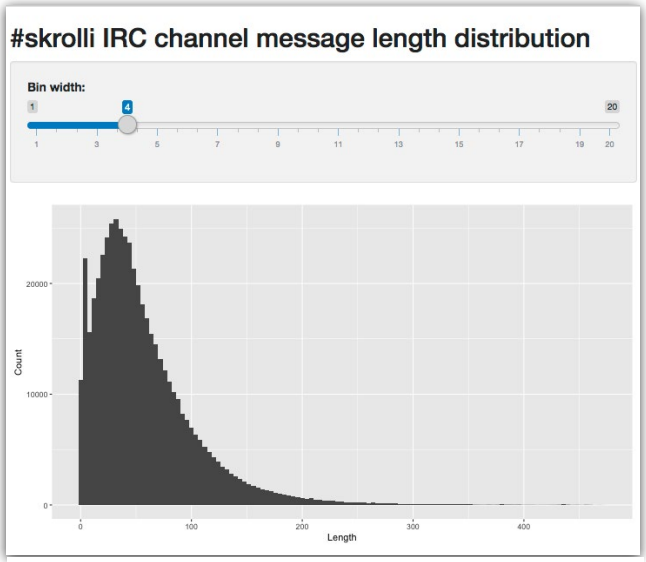


Figure 4. An interactive histogram of line length on the channel #skrolli.

As an example, let us look at the logs from the IRC channel #skrolli between 9 September 2012 and 18 July 2017 and use them to create the graph in Figure 1 that shows the frequency of specific selected words on the channel as a function of time. The R code used for plotting the graph is in Listing 2. The logs have been pre-processed with a Python script that outputs one row per each word on the channel; it includes the word and the message date, separated by a comma. This is read in on line 07 of Listing 2. In the R interpreter, the first five rows now look like this:

	date	word
1	09/09/2012	paljo
2	09/09/2012	merkkimäärä
3	09/09/2012	välilyönnit
4	09/09/2012	mukaanlukien
5	09/09/2012	ööh

Line 09 sets the resolution for the graph by rounding down the dates to the nearest month. Line 10 compares each word to different regular expressions (regexes). A new column that contains TRUE is added for each matching regex; if a match is not found, the column contains FALSE. (See Table 1.)

Next, lines 14–21 calculate the number of occurrences per month for each regex. This uses the operator "%>%", that carries the data from one function to another, similarly to the Unix pipe operator "|". This results in the following table:

	date	skrolli	assy	trump	rasp
	<date>	<int>	<int>	<int>	<int>
1	2012-09-01	209	10	0	38
2	2012-10-01	181	27	0	134
3	2012-11-01	114	28	0	58
4	2012-12-01	53	16	0	72
5	2013-01-01	55	8	1	8

Finally, the graph is plotted on lines 22–30. The melt function processes the data table into the format required for the plot function. The plus operator is used to combine the different components of the graph. (See Listing 2.)

Next, we will attempt some statistical curve fitting. Continuing the code from Listing 1, we will calculate the activity of the channel over time and fit a local regression curve. The required code is in Listing 3. Regression is handled by the geom_smooth function that accepts the applied statistical method as a parameter; currently, it is "LOESS" or Locally Weighted Scatterplot Smoothing. The end result can be seen in Figure 2.

RStudio and interactive visualisations

The most popular graphical development environment for R is known as RStudio. The software can be used either as an assistant for normal command line use or a heavier tool for project management. The user interface is fairly similar to the graphical user interface for MATLAB (see Figure 3). Compared to the command line interface, one of the key benefits of RStudio is that all available variable names and their types and values are visible. It also offers some useful tools such as a code editor and debugger as well as a profiler that can be used to find the bottlenecks in code performance.

For interactive visualisations, you can use the Shiny package that is an excellent match for RStudio. RStudio even has ready-made templates for creating Shiny visualisations. The system consists of two parts: a user interface and a server. The user interface is an HTML page that contains data views and adjustments for the parameters. The page sends the parameters to the server, which generates and returns the desired views, such as tables and graphs, for the user interface. Figure 4 contains an interactive histogram of the message length distribution on the IRC channel #skrolli that uses this system for plotting.

We will demonstrate the system by plotting an interactive histogram of the message length distribution on the IRC channel #skrolli. The server and UI side code is in Listings 4 and 5 and the end result can be seen in Figure 4. Again, the IRC logs were pre-processed in Python.

Observant readers may be wondering why we used Python instead of R for the pre-processing of data in our IRC visualisation example. This is typical in data analysis: pre-processing is done with a generic language before switching to R for data analysis and visualisation. In this case, the reason for the switch was that R is clumsy when handling strings, whereas such an operation is easy in Python.

```
01: library(dplyr)
02: library(tidyr)
03: library(ggplot2)
04: library(reshape2)
05: library(lubridate)
06:
07: df = read.csv("data.df")
08: df$date = as.Date(df$date, format="%d/%m/%Y") # Cast to date
09: df$date = floor_date(df$date, "month") # Round down to nearest month
10: df.matched = df %>% mutate(skrolli = grepl("^skrolli*",as.
character(word)),
11:                            bitcoin = grepl("^bitcoin*",as.character(word)),
12:                            trump = grepl("^trump*",as.character(word)),
13:                            rasp = grepl("^rasp*",as.character(word)))
14: df.summary = df.matched %>%
15:              group_by(date) %>%
16:              summarise(
17:                  bitcoin = sum(bitcoin),
18:                  skrolli = sum(skrolli),
19:                  trump=sum(trump),
20:                  rasp = sum(rasp)
21:              )
22: ggplot(melt(df.summary, id="date"),
23:        aes(x=date, y=value, colour=variable)) +
24:        geom_line(lwd=0.5) +
25:        scale_color_discrete("Regex",
26:            breaks=c("skrolli","bitcoin","trump","rasp"),
27:            labels=c("skrolli*","bitcoin*","trump*","rasp*")) +
28:        xlab("Time") + ylab("Occurrences") +
29:        ggtitle("Words on the #skrolli IRC channel") +
30:        theme(text = element_text(size=24,family="Georgia"))
31: ggsave("R-word-plot.pdf", device=cairo_pdf, width=16, height=9)
```

Listing 2.

```
df.totals = df %>% group_by(date) %>% summarise(count = n())
ggplot(df.totals, aes(x=date,y=count)) + geom_line() +
  geom_smooth(method="loess") +
  xlab("Time") + ylab("Number of words") +
  ggtitle("#skrolli IRC channel activity") +
  theme(text = element_text(size=24,family="Georgia"))
ggsave("R-activity-plot.pdf", device=cairo_pdf, width=16, height=9)
```

Listing 3.

	date	skrolli	bitcoin	trump	rasp
	<date>	<int>	<int>	<int>	<int>
1	2012-09-01	209	10	0	38
2	2012-10-01	181	27	0	134
3	2012-11-01	114	28	0	58
4	2012-12-01	53	16	0	72
5	2013-01-01	55	8	1	8

Table 1.

```
shinyUI(fluidPage(

  titlePanel("#skrolli IRC channel message
length distribution"),

  # Adjustment slider for histogram bin
width
  sidebarLayout(
    sidebarPanel(
      sliderInput("binwidth",
                  "Bin width:",
                  min = 1,
                  max = 20,
                  value = 4)
    ),

    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

Listing 4: UI side code.

```
library(shiny)
library(ggplot2)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {
  df = read.csv("lenghts.df")
  output$distPlot <- renderPlot({

    # generate bins based on input$bins from ui.R
    p = ggplot(df, aes(x = length)) + geom_
histogram(binwidth = input$binwidth) +
      xlab("Length") + ylab("Count")
    print(p)
  })
})
```

Listing 5: server side code.

Another negative feature of the language is the steep learning curve, since effective use requires avoiding loop structures and a thorough knowledge of the libraries. The language is not especially well suited for parallel programming, since it requires the use of separate packages that are not particularly convenient. The R interpreter has also been found to be very slow when compared to compiled languages. In the test by julialang.org, for example, quicksort was 265 times slower in R than in C. R even loses to C in its strongest area, such as the speed of matrix multiplication, although only by a factor of 1.6. For these reasons, R is not always the best option for big data analytics, for example.

However, due to the vast selection of methods, R is a good option for quickly implemented data analyses and prototype construction. Furthermore, you can use C to build the heavy calculation routines and only use R for calling the ready-made routines and visualising the end results.

In addition to R, there are also other good alternatives to MATLAB. One option is to use Python and the SciPy library, but it currently offers fewer statistical methods than R. A port of ggplot2 for Python is in the works, but it is still missing features. Julia is an interesting new product on the data analysis programming language market. It aims to offer a modern alternative to R. Julia's design takes into account the inefficiency of R and the opportunities of modern parallel and distributed computing. Julia also has hundreds of software libraries, including a version of ggplot2.

Compared to R's vast selection of libraries, however, the range remains fairly limited. Therefore, R remains a viable language and a good addition to the toolbox of any programmer interested in data analysis.

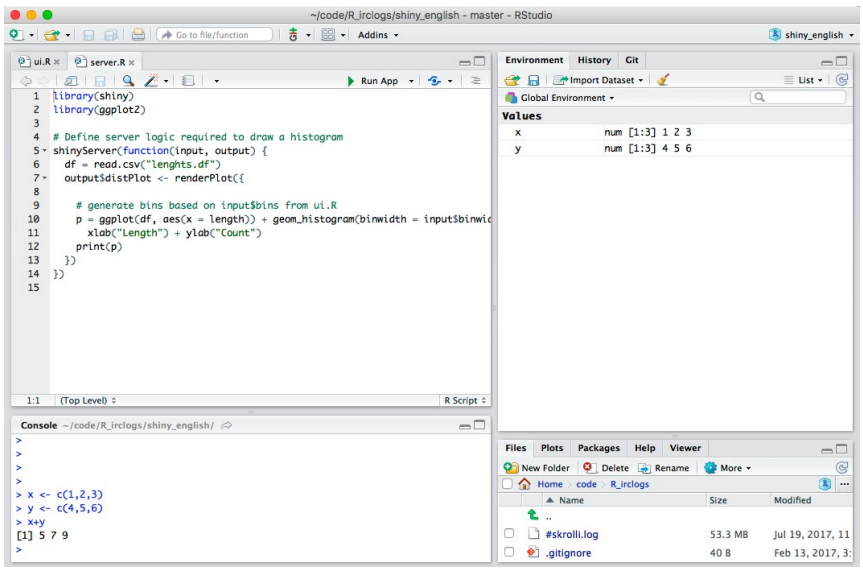


Figure 3. RStudio IDE.

Story Ville Jouppi
Pictures Ville Jouppi
Toni Bratincevic
Henrik Erlandsson

System

Dust off your Amiga 500

Were you cleaning up your closet when you came across your good ol' Amiga 500? Did you find yourself sitting with your friends and reminiscing about the golden times when the hardware was great and the games were hard? Do you think you could now totally own the Oldskool demo compo at Assembly?

Regardless of whether you are now looking at your old trusty workhorse or a previously loved eBay purchase, your mind will probably be full of questions. This article reviews some of the more common failures and problems that you can expect with a piece of classic hardware.

A large budget is not required, but if you have money to spend, there is no upper limit, of course. The cheapest alternative is to leave out all the expansions and to use floppy disks for loading software. However, if you are looking to do anything more serious, mass storage that is faster and larger than floppies is a necessity. Memory will also be in short supply if you aim to venture outside of feeding game disks into the drive.

The belly of the beast

The very first thing you should do is open up the Amiga and take a glance at its insides. Remove all the screws at the bottom of the machine, except for the three countersunk screws below the disk drive. After this, you can simply lift the top part off. There are clips on the left and right sides of the case, near the upper edge of the keyboard; these may be tight if the Amiga has never been opened before.

Taking the top off usually reveals a dull sight: the keyboard, disk drive and a large metal shield. Remove the keyboard connector from the motherboard by pulling it upward. Note its orientation. The black wire should be on the left. Lift the keyboard to one side. You can lightly tap on the keyboard to remove dust from between the keys, or use a can of pressurised air. A vacuum cleaner will also work, provided that you use a low, weak power setting.

After this, it is time to tackle the RF shield. The shield is fastened with four screws (in the front section and on the cover for the Amigabus slot on the side) and a set of small metal clasps. Remove the screws and straighten the clasps. You can now lift the shield off.

With the shield out of the way, you will see the motherboard. The first thing you should do is to take a look at the corner of the PCB, next to the floppy drive, and check the revision

Identifier for motherboard revision 6

number. Revisions of 6 and above are preferable, 5 is something you can live with. If the corner has no revision number, the motherboard is revision 3. It is mainly of collector interest due to its age and instability. Revisions 4 and 7 are very rare and you might want to offer them to a collector. They are not worth a fortune, but chances are you will sell it for more than you paid for it.

Some of the very last A500s are revision 8, which means that they are actually dumbed-down versions of the A500+. You can turn one into a full A500+ by adding a few components. If this is something you might be interested in, I recommend that you take a look at the schematic diagrams and online photographs of different motherboards in order to find the differences.

In the nick of time

At this point, you should take a critical look at all of the electrolytic capacitors – the bean-can like, plastic insulated two-prong components that stick out of the motherboard. If there are any deposits below the capacitor or if the solder joints nearby have turned green, the capacitor has failed and it needs to be replaced. You need to clean up any spilled electrolyte to stop it from corroding the motherboard. In the larger capacitors, failure may also cause the top to bulge out.

Finally, you should see if the machine has a memory expansion with a real-time clock and battery. If the memory expansion is Commodore's own A501, there is a 100% chance that the battery has failed and leaked onto the RAM/RTC module. You can recognise the A501 by its enclosure inside a large wedge-shaped RF shield that requires a soldering iron to open.

If the original battery is still in place, you must remove it before connecting the power in order to avoid leaks. Small side-cutting pliers are great for the removal: simply cut the pins and lift off the expired battery. If the battery has already failed, you need to clear off the electrolyte in order to avoid further damage. Unsweetened lemon juice works well in neutralising the electrolyte, and final cleaning is best done with isopropyl alcohol.

You can leave out the battery if you only intend to use the Amiga for gaming. The real-time clock will no longer show the correct time, but this should be a very minor nuisance. Users of utility software should replace the battery, since file timestamps that make sense are useful – and you might occasionally glance at the desktop clock. You can also replace the rechargeable battery with a coin-type lithium battery and diode. The diode is installed between the positive battery holder terminal and the motherboard, with the conducting direction towards the motherboard. If you feel that this is above your level of competence in electronics, a ready-made replacement kit is available from Amigakit.

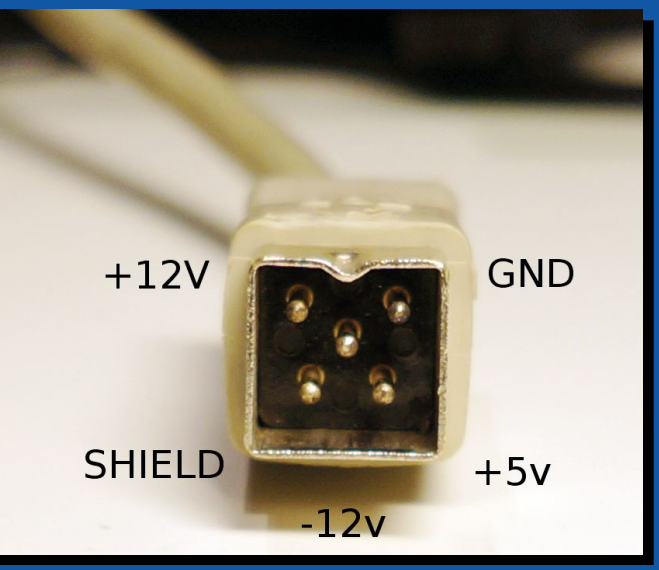
Pictures on the screen

The Amiga is a product of the era when home computers were connected to televisions. Therefore, the video signal from a PAL system more or less follows the 288p standard. This means a vertical refresh rate of 50 Hz, whereas the horizontal frequency is 15 kHz. Instead of the normal interlaced TV signal, subsequent frames are sent on the same scanlines as the previous ones, which results in a genuine refresh rate of 50 frames per second.

Even modern TVs can handle a traditional 50 Hz PAL image, but a

progressive signal may be slightly more challenging. Many modern televisions perform heavy image processing and force interlacing removal even on a progressive signal. This means that every other frame is not displayed, and games that feature smooth scrolling, such as *Pinball Dreams,* seem jerky.

You can double the 15 kHz horizontal frequency with a device known as a "scandoubler" or "flickerfixer". An example of such a product is the Indivision that can still be purchased new. However, unless your VGA monitor can sync to a 50 Hz vertical refresh rate, scrolling will not be completely smooth. ViewSonic and Sony monitors from the early 2000s can do this. I recommend that you study the refresh rate tables in the user manual before buying a second-hand monitor.



The power connector on an Amiga 500.

You can also use an upscaler, such as Micomsoft's X-RGB Mini Framemeister or the Open Source Scan Converter. They are pricey, but can convert any type of progressive signal and output to modern HDMI.

Of course, the best possible compatibility can be achieved with an old CRT television and the SCART connector. RGB SCART cables can still be bought new. Old monitors from the era, such as the Commodore 1084 or the Philips CM8833, also work reliably and create a cosy retro atmosphere.

Powering up the system

If your power supply is missing or broken, you can purchase a replacement or replace the low voltage side connector on an existing one.

The voltage and power requirements on an A500 are as follows:

* +5VDC 3 A
* +12VDC 1 A
* -12VDC 0,1 A

In an expanded Amiga, you would do well to provide more amps on the 5-volt line. The 12V line is mostly required for spinning hard drive motors. Nowadays, most people use flash cards as mass storage, and they largely operate on five volts. The negative 12V line is provided for sound and the RS232 line, and it is mostly used as a reference voltage.

Tapping an old AT or ATX power supply for these voltages is fairly easy, and there are ready-made adaptors available online. There is no need to overdo it with the wattage, however. An Amiga does not consume that much power, after all, so if your power supply is too large, the load from the Amiga might not be enough for it to regulate voltages properly. Furthermore, at some point, the 3.3-volt rail became more important than the 5-volt one in the PC world, so older ATX power supplies with 250–300 W outputs are the safest option.

The Amiga uses a square power connector that is difficult to find new. The easiest thing to do is to recycle one from a broken power supply. You can commonly find

No signs of life?

Once the monitor, power supply and mouse have been connected, the machine should come to life after you flick the switch. If this does not happen, see below for additional instructions.

Keyboard
When the Amiga starts up, the Caps Lock LED will normally turn on for a moment and then switch off. Sometimes, the LED will continue blinking and the keyboard will not respond. In this case, the first step is to measure the supply voltages. Occasionally, a broken CIA chip may also be the cause.

The keyboard will try to communicate the error type with the number of blinks:

1. Incorrect ROM checksum
2. RAM test failed
3. Watchdog timer error

If the supply voltages are correct, you have tried switching the CIAs around and the problem will not go away, the keyboard control chip has died and you need to replace the keyboard (or its controller chip in the top corner).

Screen freezes in a specific colour
The Amiga can also use colour codes to signal problems encountered during the initial tests.

The most common colours are red, yellow and green. The following explanations mainly apply to the Amiga 500.

* Red – Kickstart ROM checksum error. Check the pins and sockets of the CPU and Kickstart ROM for oxidation. Clean them, if necessary.
* Yellow – The CPU executed code that threw an exception while booting. This is most commonly caused by a defective expansion. Remove all expansions and try to restart the machine.
* Green – Chip RAM error. Chip memory

is the area of the Amiga's memory that graphics and sound chips can access. A green screen is commonly a symptom of contact problems in the pins or the socket of the Agnus chip. However, it may also occur if one of the memory chips on the motherboard or the trapdoor memory expansion is defective.

A word of warning to those wishing to solve the green screen problem: Always use a dedicated extractor when removing PLCC chips. Using two screwdrivers will result in a broken PLCC socket, since these old chips are tight and their sockets become brittle over time.

Floppy drive does not work

There are many more A500 motherboards in the world than there are working, compatible floppy drives.

PLCC extractor.

The first thing to try is a drive cleaning disk with a couple of drops of isopropyl alcohol. If you do not have a cleaning disk, you can also clean the heads with a cotton swab dipped in isopropyl alcohol. This requires precision and a firm but gentle touch in order to prevent the misalignment of the drive heads.

While the drive is open, you should also remove all the dust and solidified grease from the read/write head slide bar. Use a running oil, such as sewing machine oil, to lubricate the slide bar. Only use a small amount of oil in order to avoid spills. Bear in mind that WD-40 and its equivalents are only suited for loosening rusty nuts, not long-term lubrication.

Some Panasonic drives have leaking surface mounted capacitors. In most cases, you can bring the drive back to life by replacing the failed capacitors. The most common failure point can be found at the front of the drive, next to the flywheel.

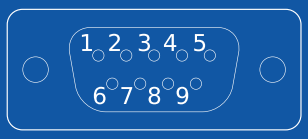
In Chinon-branded drives, the most common problem is that the write protection and disk detection switches at the front operate intermittently. You can try dropping a small amount of isopropyl alcohol into the switches and pressing them down with your finger for a short while. For the best results, you can break down the switch and carefully clean the contacts inside — but I cannot recommend this unless you are an expert clockmaker. The small parts and springs are easy to lose, and reassembling the switch is also quite tricky.

Fortunately, working floppy drives can still be found for sale online. It is also possible to modify PC drives for Amiga use; a simple web search will produce a tutorial.

Mouse is dead, joystick brings no joy

If the mouse appears defective, the very first thing to do is to borrow another mouse for testing. If even the borrowed one does not work, measure the voltage on the mouse port at the back of the machine. Does pin 7 have 5 volts? If not, one of the passive components near the mouse port might be the cause. The schematic diagrams are available online, so all you need to do is follow the traces and try to find the component where the voltage no longer passes through.

If the supply voltage is close to five volts, you can try to replace the Denise chip with a known good version. However, a much more common problem is a burnt out multiplexer 74LS157 at



Mouse port connector.

U15. Unfortunately, U15 is not socketed, so replacing it purely for the sake of experimentation is a quite a lot of work. The part is cheap, though, so even if you replace it for nothing, the financial loss will not be great. Remember to use a socket for the new chip.

Joystick directions do not require +5V. If the directions do not work, U15 is again the most likely culprit.

If the left mouse button or the joystick fire button does not work, the fault is in the CIA chip at U7. Interestingly, the first fire buttons of both game ports are the only function of the Amiga's CIA that is related to game controllers, even though replacing this chip is by far the most commonly suggested solution to all joystick problems.

The right and middle mouse button and the second and third fire button on the joystick are read from the potentiometer input on the Paula chip. If there are any problems with these buttons, the cause is most likely the Paula chip at U3.

used power supplies at online trading sites, and retailers naturally stock new power supplies and adaptors.

Mouse ran away?

Machines stored in cupboards have often lost their mice. Perhaps the most cost-effective solution is to purchase one of the several commercial or hobbyist-built PS/2 or USB mouse adaptors. Many USB mice can still switch to PS/2 mode when used with a passive adaptor.

Retailers still sell new Amiga compatible mice and used ones frequently appear on sale online. If you still have your old broken "tank mouse", you can fit a new logic board that turns it into an Amiga compatible laser mouse.

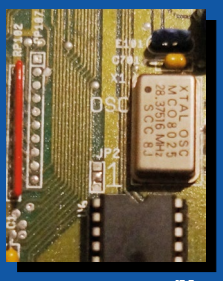
One-meg chip? With or without fish?

A normal A500 has 512 kilobytes of chip RAM on the motherboard, and putting 512 kilobytes of "slow fast RAM" in the trapdoor slot was a common expansion. Chip memory is available to the custom chipset and the CPU, whereas only the CPU can access the fast RAM.

In the 1990s, every Amiga user worth their mettle modified their machine to address an entire megabyte of chip RAM. Now you could easily play huge ProTracker modules and have several screens open at once! This is still a worthwhile modification for productivity purposes, since you can never have too much chip RAM. Gamers using floppy disks need not bother, how-

1Mb Chip memory

If you wish to perform the modification, the first thing to check is the model number on your Agnus chip. The model number needs to be 8372 to allow for a full 1 MB of chip RAM. The 512-kilobyte trapdoor expansion also needs to be fitted.



JP2.

The modification itself is easy. The top address pin on the Agnus is switched to point to address A19 on the address bus, and the EXRAM signal to GARY is cut in order to prevent the same memory from showing up in two memory ranges.

JP7.

The address bus modification is done by changing the jumper JP2. In Rev6 motherboards, you can switch the EXRAM signal off by cutting the trace between the middle and bottom pins of jumper JP7A. In case of a Rev5 motherboard, you can lift GARY of its socket and bend the EXRAM pin outward, or if you have a third-party memory expansion, turn its power switch to the OFF position.

When the middle and front pins of JP2 are connected, only 512 kilobytes of chip RAM is visible; the middle and rear pins will give you 1 MB. In order to achieve 1 MB, you need to cut off the EXRAM signal, since the memory expansion is added to the chip memory. In half-meg mode, you can select whether or not the memory expansion is visible.

Red: cut, green: solder.

You can change the position of JP2 simply by cutting the trace on the motherboard between the middle and front pads and by soldering the other two pads together. Instead of a permanent modification, I recommend fitting a DPDT switch that can simultaneously modify the position of JP2 and the state of the EXRAM signal. There is ample space for a switch on the rear panel next to the joystick ports.

In Rev5 motherboards, you most often need to update the Agnus chip. When doing this, you need to remember to isolate the PAL/NTSC pin 41 on the Agnus from the socket with a piece of tape; otherwise, the machine will always start up in NTSC mode. My earlier warning is also worth repeating here: always use a PLCC extractor.

In addition to this simple 1-megabyte chip modification, there are also 2-megabyte expansions that fit in the socket of the Agnus chip. If you want one, keep your eyes peeled on the Web — however, due to their scarcity, these expansions command a fairly high price.

More detailed instructions for the modification can easily be found online. It should also be noted that if you have a Rev6 motherboard and no trapdoor expansion, you can solder DRAM chips and their filter capacitors directly in the empty slots on the motherboard. If you decide to take this approach, I recommend that you study the schematics. Note also that the empty memory slots on the motherboard overlap the trapdoor expansion; if you have a full megabyte on the motherboard, do not place an expansion in the trapdoor.

ever, since most games running on A500 can make do with 512 KB of chip RAM. Some also need an additional 512 KB of fast RAM.

Can I kick it?

The Amiga's Kickstart chip contains about one half of the operating system. People coming from the IBM PC world often see it as a BIOS replacement, but they are not directly equivalent. When the Amiga boots up and the insert disk prompt is shown, multi-tasking and AmigaOS are already running. When BIOS shows the insert disk message, an IBM PC has no operating system in memory; it only has a low-level start-up routine that can load a single sector from the beginning of the disk.

Users of floppy games need not concern themselves with this, since version 1.2 or 1.3 is perfectly good and offers the best compatibility. Power users opt for the latest version, 3.1 (rev 40.63). A lot of productivity software requires at least version 2.04 (rev 37.175), but there is no harm in updating straight to 3.1. If anything, it is more compatible with old games than 2.04.

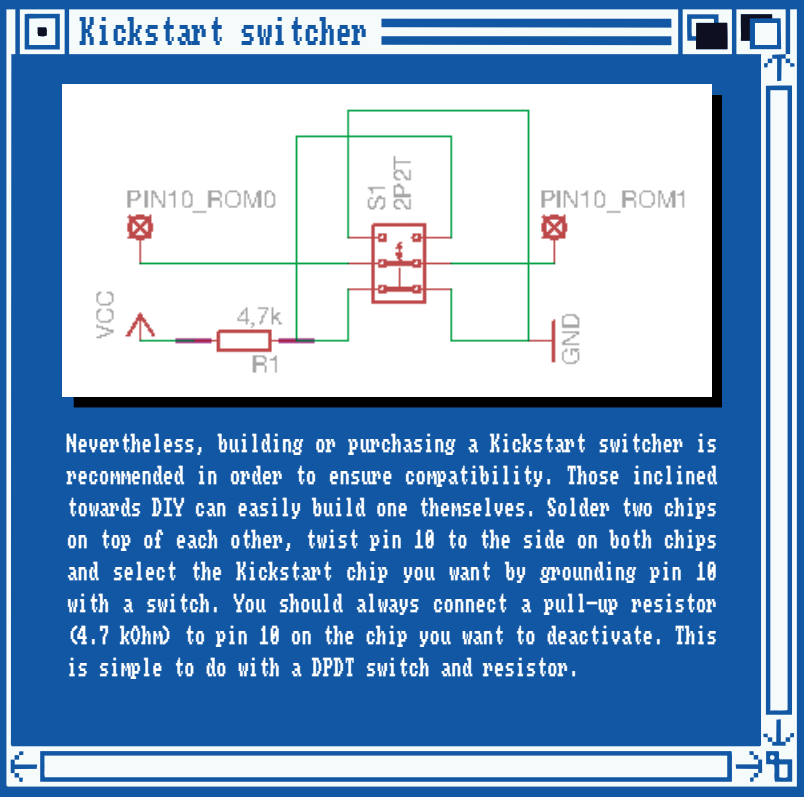
Rev5 and older motherboards have their Kickstart sockets erroneously connected. The top address line is on the wrong pin. Kickstart 3.1 is nearly always supplied on an EPROM chip, so the following changes are required. Twist pin 31 up to prevent it from contacting the socket. Use a jumper to connect pin 1 on the chip to pin 31 on the socket. Pin 31 on the chip, which was twisted out previously, is connected to pin 21. The first jumper fixes the erroneous address line, the second jumper ensures that the chip is working in 16-bit mode. These changes are not required on Rev6 and newer motherboards.

Media for the masses

Mass storage is not a concern for floppy enthusiasts: the game will load once you pop in the floppy.

Gotek floppy drive emulator.

nally loaded from floppy "on the fly" and runs them from the hard drive. At the same time, WHDLoad introduces compatibility fixes and allows even old games to run on expanded Amigas. In practice, WHDLoad requires at least a couple of megabytes of RAM and a hard drive to work.



Power users cannot really live without a hard drive or flash storage of some sort. However, even players should consider expansions, since WHDLoad makes the system much more comfortable to use.

WHDLoad is a compatibility framework that converts games originally loaded from floppy "on the fly" and runs them from the hard drive. At the same time, WHDLoad introduces compatibility fixes and allows even old games to run on expanded Amigas. In practice, WHDLoad requires at least a couple of megabytes of RAM and a hard drive to work.

Replacing the disk drive with an emulator is also a modern option. The first, lower-cost option is the Gotek floppy emulator that can be reflashed for Amiga compatibility. This way, you can place your disk images on a USB stick and will not need to store or write a large number of floppies. Goteks are available on eBay and Amiga sales forums. They can use either the earlier Cortex firmware or the later and more advanced HxC version that you need to purchase separately. You simply copy your image files onto a USB stick and Gotek will start with a menu where you can select the disk image to boot. The buttons on the side of the emulator can be used for switching disks.

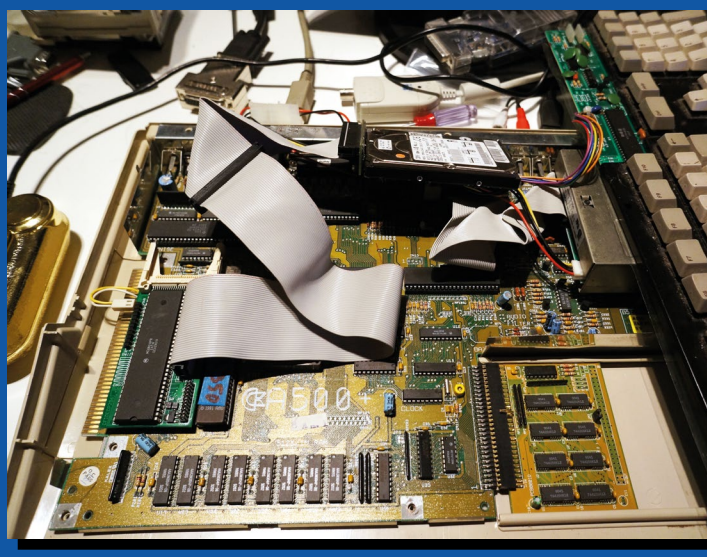
The HxC Floppy Emulator uses SD cards. It was originally designed as a kit, but ready-made versions are available. It fits a host of other systems alongside the Amiga. You can also use your computer as a storage medium by purchasing the USB version. The HxC is also available with a screen that allows you to choose the image that you want to insert.

The harder they drive

Floppies are nice, but all real power users have hard disks. Most hard drive controllers for the A500 use the AmigaBUS connector on the left side of the machine. Back in the 1980s, SCSI was the best way to connect mass storage devices to home computers, so most hard drive controllers for the A500 use SCSI.

The best SCSI controllers for the A500 are the HD8+ from GVP and Commodore's own A590, since they have DMA support. DMA, or Direct Memory Access, means that they can move data directly into the computer's memory without CPU involvement. The rest are more or less equal and use the processor to move bits around.

If you choose something other than the GVP or A590, ensure that your controller supports autoboot and RDB partition tables. Without autoboot, your future will be filled with boot floppies and RAM drives, and if your controller does not support RDB, you can only use the partitioning tool from the manufacturer. In all likelihood, it will not take kindly to modern file systems and gigabyte-class hard drives. Old hard drive controllers appear for sale every now and then, but their prices tend to fluctuate.



Mika Leinonen's IDE interface.

Mika Leinonen from Finland and Kipper2k from Canada have designed expansions that add an A600 compatible IDE bus to the CPU socket of the A500. Kickstart 3.1 includes a driver for the IDE bus, which makes these cards an attractive option.

My first recommendation for the hard drive file system is PFS3 AIO; it is simple to install, supports large hard drives and works with nearly all CPUs and Kickstart versions. When using the Kickstart IDE driver, the maximum hard drive size in PFS3 AIO is 8 gigabytes.

Accelerators: Feel the need for speed

An accelerator board replaces the Amiga's CPU with a faster version. A pattern has already emerged with the other expansions, and it holds true here: to play games from floppy, you should stay away from accelerators due to compatibility reasons.

Then again, power users need all the additional oomph they can get. An accelerator is also useful for WHDLoad, since some installs require a faster CPU and a few megabytes of RAM.

Old accelerators come in two types: Ones that connect to the socket of the 68000 CPU and ones that use the AmigaBUS connector on the side. Historically, the AmigaBUS ones were better. Memory is the main problem with old accelerators using the 68000 socket. You can only access a couple of megabytes or the memory is not autoconfiguring, in which case a part of the operating system may be loaded into slow chip memory at boot. Furthermore, the DMA capable SCSI controllers on the AmigaBUS cannot transfer to the 32-bit memory of the accelerator.

The A530 from GVP uses the AmigaBUS interface. It is a rare expansion but probably the finest "classic" A500 accelerator. The fast 68030 CPU, 32-bit memory and DMA enabled SCSI controller allow it to outperform even the Amiga 3000 in many cases. The SupraTurbo28 is among the slowest A500 accelerators, but it contains cache memory that speeds up the 28 MHz CPU somewhat. SupraTurbo also requires true fast RAM in order to be useful.

Some new expansions are also available. The ACA500plus from

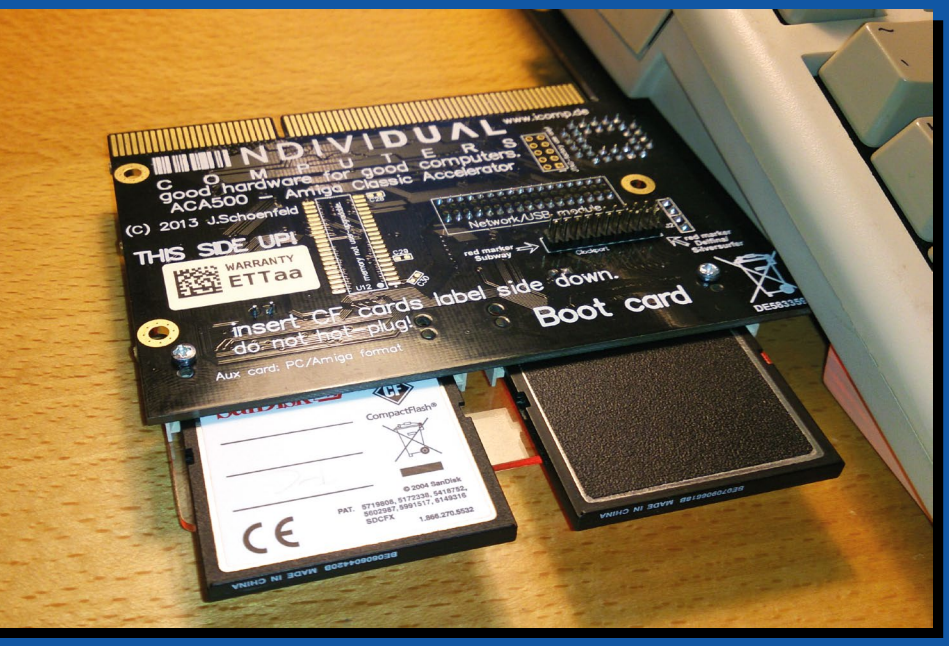
Individual Computers is an interesting option for light productivity use or WHDLoad gaming. It includes a 68000 CPU at 14 MHz and two CompactFlash slots, one for Amiga format cards, another for the PC's FAT format. It also comes with a slot for A1200 accelerators and many other features, all at a cost-effective price. Those with a larger bankroll may be tempted by the Vampire 500. It is an FPGA based accelerator that contains an SD card slot, an HDMI connector and more computing power than you have ever seen in an Amiga 500. The Vampire replaces the 68000 CPU with a software implementation that can even turn the A500 into an AGA machine. However, availability is scarce and prices were fairly high at the time of writing.

When used with only a memory expansion and a hard drive, WHDLoad will also benefit if you upgrade the processor to a Motorola 68010. The movable vector register on the 68010 allows you to exit games without resetting the machine. The processor upgrade is simple, since the 68010 fits directly on top of the old CPU.

SupraDrive 500XP + SupraRam 500RX 8MB.

If you only want to use WHDLoad, a simple memory expansion may be sufficient. They are available as AmigaBUS versions and ones that connect to the CPU socket. Kipper2k's expansions are the best option in this case, since they have a memory expansion and a Kickstart 3.1-compatible CompactFlash slot on the same board.

Memory expansions installed on the AmigaBUS or the CPU socket provide real, 16-bit fast RAM; this means that the CPU does not need to wait for the custom chip DMA when running code from real fast RAM. This alone results in a power increase of some 3% when compared to a machine with only chip RAM or a trapdoor expansion.



ACA500.

Transferring files

After all this hardware analysis, tinkering and expansion, a look at software is also in order. After all, staring at the Workbench prompt on an upgraded machine has very limited appeal.

It is exactly this chicken and egg problem that new Amiga owners face. An Amiga will not boot off a PC disk, and a PC cannot write Amiga compatible floppies without additional hardware. Not to mention that very few modern PCs have any kind of floppy drive.

Gotek and HxC users have it easy: they simply need to copy the disk images to a memory card or USB stick and be done with it. ACA500plus users can install WorkBench 3.1 on a CF card from the accelerator's memory and use the second card slot to read PC format cards.

For floppy drive users, the RS232 bus is the most cost-effective solution. For RS232 transfers, I recommend the commercial Amiga Explorer or the community-built Hombre package. Amiga Explorer includes clear

instructions, and you only need the Workbench disk to get started. Hombre requires some more effort, but it includes several good tools for data transfer.

You can make your PC write Amiga floppies by purchasing a KryoFlux, Catweasel or Supercard Pro. These are programmable floppy drive controllers that use standard PC floppy drives. They allow for reading and writing disks for many PC incompatible devices. All of these devices are somewhat hobbyist oriented, and their price may also scare some people away. However, if this sort of thing interests you, I can recommend the KryoFlux. It has proved to be worth the price when preserving old software. The KryoFlux connects to the USB interface and comes with good software.

If the Amiga uses a hard drive or memory card that you can connect to a household PC, you can use the WinUAE emulator to mount the drive on an emulated Amiga or even boot the emulator from it. Copying files inside the emulation is simple when you mount a folder from the PC side as a virtual hard drive. Partitioning the drive or CF card and installing the operating system and software are also easy to do with WinUAE. I use it quite a lot when setting up a CF card for an Amiga

Other fun things

An additional disk drive will help players and power users alike. Several games support a second floppy drive and disk-swapping is reduced.

Coders may also want to consider purchasing an Action Replay 3. It contains a machine code monitor, different rippers and disk access functions at the press of a button. The ACA500plus contains an Action Replay Mk II and III.

Musicians may be interested in a parallel port sampler that you can use to record sounds for ProTracker, for example.

To play around with analogue video, you can purchase a genlock that replaces the background colour with an external video feed. This way, you can create your own subtitles or logos.

And more...

This article was only a small sample of what expanding the Amiga 500 can offer. There will never be enough room to show every option, but I hope I have piqued your interest in what you can achieve.

Reading the Amiga forums is recommended, as they contain a wealth of on-topic and off-topic information.

A floppy drive will serve you well for gaming as long as you can write the disks somehow. Used Amiga disks are available for sale, and you only need a few to get started.

An expanded Amiga comes to its own in making music, drawing or programming practice. Even then, you need not buy everything at once; some extra memory and a hard drive will get you started.

Links

Trading sites:
- www.amibay.com (active, but with very strict rules)
- www.amibench.org (more liberal but fairly barren)
- ebay.de, ebay.co.uk

Forums:
- eab.abime.net
- www.amiga.org
- saku.bbs.fi (In Finnish)

Resellers:
- www.gentle-eye.fi (The last Amiga reseller in Finland)
- www.vesalia.de
- www.ggsdata.se
- www.amigakit.eu

Information on accessories:
- www.bigbookofamigahardware.com
- amiga.resource.cx

Software:
- www.aminet.net
- www.whdload.de

Floppy drive emulators:
- hxc2001.free.fr/floppy_drive_emulator
- cortexamigafloppydrive.wordpress.com

Data transfer:
- wiki.abime.net/file_transfer/hombre
- www.amigaforever.com/ae

Build a USB joystick adapter from an Arduino

Would you like to use your favourite game controller with an emulator? Or would you like to learn some Arduino coding? This little project will help you with both.

Story and images by Jarno Lehtinen

An Arduino based on the ATmega32u4 microcontroller can fairly easily be turned into a USB joystick adapter. This adapter allows your old joystick to work on almost any modern device with a USB port. In this story, we will be building an adapter for Atari type joysticks using an Arduino Pro that can be found on eBay for less than $5. This adapter allows for connecting many game controllers from the 1980s and 1990s that use a 9-pin D connector.

All you need is a computer for programming the Arduino – you can use Windows, Linux (x86 or ARM) or OS X –, an ATmega32u4 based Arduino, a microUSB cable, a 9-pin male D connector and some wire.

Different joystick implementations

There are many different ways for connecting game controllers and processing the signals. The simplest one is the Atari joystick, where one switch corresponds to each direction or fire button. The positive side of this is simplicity; the negative side is that you need just as many wires and pins on the connector as there are directions or fire buttons. The joystick requires no voltage to operate and the signal voltage does not matter. You can find instructions for building an Atari joystick in Skrolli 2016.1E.

The control pad for the Sega Mega Drive (or Genesis in North America) works similarly to the Atari controller, but one of the pins is used to select between different modes of operation. This saves on wires and pins, since each wire can relay two different buttons or functions. A six-button Mega Drive pad uses 3 modes.

On the Nintendo NES and SNES, the controller sends the state as a unidirectional serial bit stream. The console sends a clock signal and reads the data one bit at a time.

The PlayStation 1 and 2 controllers are based on bidirectional serial bit streams, where data is sent and read 8 bits, or 1 byte, at a time. The clock signal is sent by the console.

The Nintendo 64 and GameCube controllers are the most challenging from Arduino's perspective, since reading them requires exact timing. Cycle-exact timing is only possible on the Arduino when machine code is used. Many example programs found online assume that the Arduino is operating at a clock rate of 16 MHz, and will not work on an 8-megahertz Arduino.

There are also controllers that are based on an analogue signal. From Arduino's point of view, these may be considered to be potentiometers. For example, the joystick on the Apple II and IBM PC and the paddles on a C64 are analogue controllers. Since the Arduino contains analogue inputs, even analogue controllers can be read easily.

Arduino

For this project, the most important thing is that your Arduino has the ATmega32u4 microcontroller. They can be found on eBay with the search term "ATmega32u4", for example. Product names include "Leonardo" and "Pro Micro". The ATmega32u4 is available in both 5V and 3.3V versions. Most controllers run on five volts. The dif-

ferent voltage versions also have different clock frequencies (16 MHz and 8 MHz), which means that software requiring precise timing will not work correctly between them. Most of the example programs available online assume that the Arduino is operating at 16 MHz. The adapter built here works with both versions. There are also versions of the Arduino that do not have a physical USB port, so ensure that yours does.

A programmed Arduino will be visible as an HID compliant USB device. For example, Windows Control Panel will recognise it as an "HID-compliant game controller".

You can also build the adapter without soldering. For this, you need to purchase an Arduino that already has the pin headers attached, some connecting wire and a 9-pin D connector with a breakout board that you can connect wires to without soldering.

Getting to work

Connect the 9-pin male D connector to the Arduino. The pin order can be seen in the diagram. You can use any pins by changing the pin definitions in the program listing, but using pins 0 and 1 is not recommended. They are used as a serial port, and even though the example code does not use the serial port, it is often used for debugging purposes during development. For example, you can program the Arduino to write the joystick state to a serial port, which allows it to show in the serial port monitor of the Arduino IDE.

Download and install the Arduino IDE on your computer from www.arduino.cc/en/Main/Software. Tell the Arduino IDE the model you are using by selecting, for example, Tools→Board→Arduino Leonardo (this selection also works for the Arduino Pro Micros available on eBay). Select the COM port for the Arduino under Tools→Port. If you are unsure about the port, start with the highest COM port number. In many cases, it is the correct one.

Next, we need to download and install ArduinoJoystickLibrary. Go to github.com/MHeironimus/ArduinoJoystickLibrary and click "Clone or download" and "Download ZIP". From the downloaded ZIP file, unpack the folder "ArduinoJoystick-Library-master" to the library folder on the Arduino IDE. On Windows, the library folder is %HOMEPATH%\Documents\Arduino\libraries, and on Linux it is ~/Arduino/libraries. The files are in the right place if the following files can be found:

On Windows: %HOMEPATH%\Documents\Arduino\libraries\Joystick\Joystick.h and Joystick.cpp

On Linux: ~/Arduino/libraries/Joystick/Joystick.h and Joystick.cpp.

Start a new project in the Arduino IDE and copy and paste the enclosed listing from the magazine's PDF version or the Skrolli website. Click the Upload button (round button with an arrow pointing to the right located in the top left corner of the Arduino IDE). The Upload button compiles the source code into code that the Arduino can understand and uploads it to the Arduino's internal flash memory. At this point, the COM port number of the Arduino may change on Windows, as the joystick library reactivates the Arduino as a USB-HID device. Change the COM port setting, if necessary, and click the Upload button again.

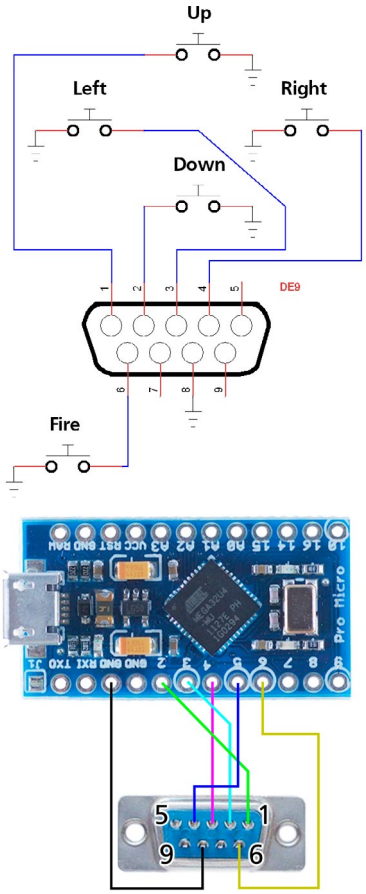
If everything went correctly, your adapter should now be ready. You can test it with the Game Controllers setting in Windows Control Panel, or on Linux with jstest /dev/input/js0.

Finishing touches

Some controllers require small changes in the code if you want to use a second fire button. For example, the Sega Master System pad is compatible with the Atari connector, but if you want to add a second fire button, add another button on pin 9 of the D connector.

The joystick built in 2016.1E will also work with this adapter. Note that this joystick has the ground pin on pin 9 of the D connector. Connect pin 9 on the D connector to GND on the Arduino. You can use the second fire button by adding it to pin 7 on the D connector.

In addition to a joystick, the Arduino will also operate as a keyboard and mouse. If you intend to use the adapter on an Android mobile device, for example, turning the Arduino into a keyboard is more compatible. The keyboard library is built in to the Arduino IDE, and instructions for using it are available at www.arduino.cc/en/Reference/MouseKeyboard. The example program presented here is available as a keyboard version at github.com/mcgurk/Arduino-USB-HID-RetroJoystickAdapter/tree/master/Tutorial.



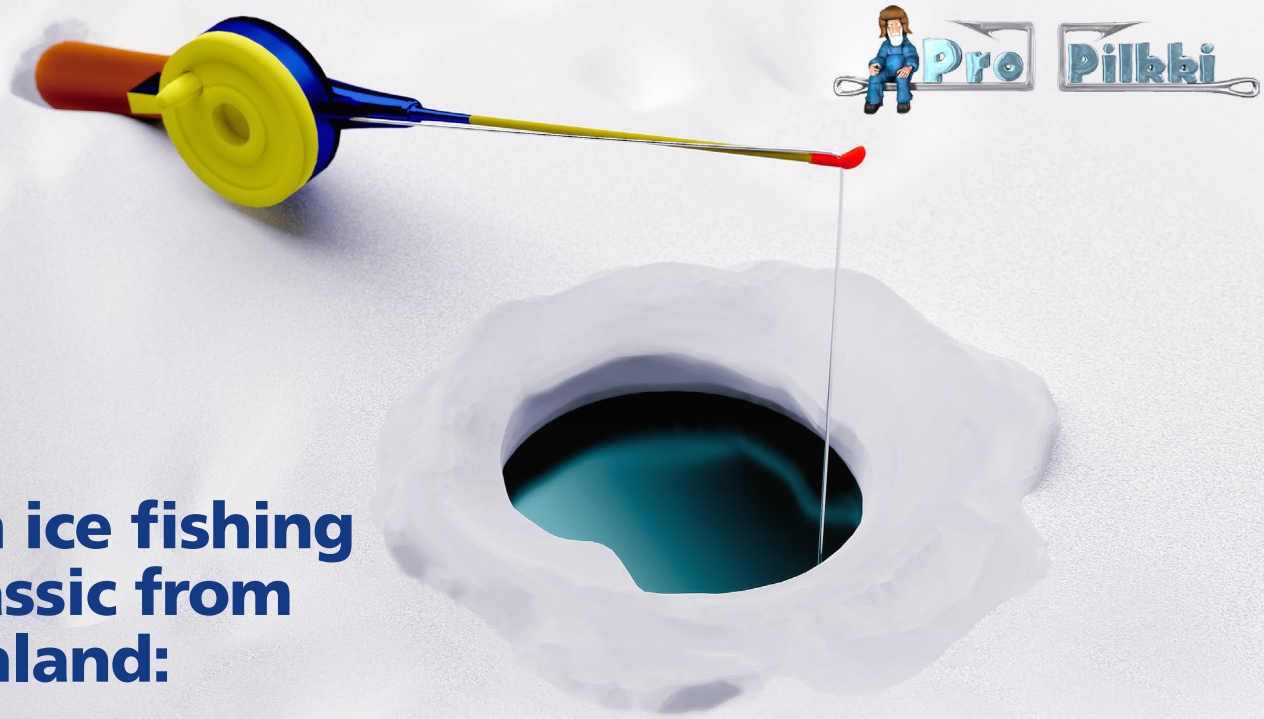
You can also build the adapter from a regular Arduino UNO that contains an ATmega16u2 microcontroller in addition to an ATmega328p. Normally, the 16u2 manages serial port communication between the computer and the Arduino, so the 16u2 needs to be separately programmed as an HID device. The PCB of the UNO does not have enough I/O contacts to build an Atari adapter with the ATmega16u2 alone, so the ATmega328p needs to be reprogrammed to read the controller state and send it to the 16u2. Instructions are available at github.com/NicoHood/HoodLoader2.

You can also build a software USB port into the Arduino by using three resistors and two Zener diodes. The V-USB library and its documentation can be found at www.obdev.at/products/vusb/index.html.

Additional information and ready-made software for different controllers is available at github.com/mcgurk/Arduino-USB-HID-RetroJoystickAdapter.

```
/*
 * Atari joystick USB adapter for Skrolli Magazine
 * by Jarno Lehtinen
 */

#include "Joystick.h"

// Define which IO pins on the Arduino the 9-pin D
// connector is wired to.
#define UP 2 // D connector pin 1 (up)
#define DOWN 3 // D connector pin 2 (down)
#define LEFT 4 // D connector pin 3 (left)
#define RIGHT 5 // D connector pin 4 (right)
#define BUTTON 6 // D connector pin 6 (fire)
// You also need to connect pin 8 (ground) on the D
// connector to GND on the Arduino.

// Keep track of the previous joystick state in order to
// check if the new state is different. Since we are
// using the internal pull-up resistors on the Arduino,
// an open switch gives a value of 1 and a closed switch
// grounds the signal, resulting in a value of 0 (1 =
// off, 0 = on)
byte lastUP = 1;
byte lastDOWN = 1;
byte lastLEFT = 1;
byte lastRIGHT = 1;
byte lastBUTTON = 1;

byte newUP = 1;
byte newDOWN = 1;
byte newLEFT = 1;
byte newRIGHT = 1;
byte newBUTTON = 1;

void setup() {
  // Define the used pins as inputs and activate the
  // pull-up resistors.
  pinMode(UP, INPUT_PULLUP);
  pinMode(DOWN, INPUT_PULLUP);
  pinMode(LEFT, INPUT_PULLUP);
  pinMode(RIGHT, INPUT_PULLUP);
  pinMode(BUTTON, INPUT_PULLUP);

  // Activate the USB joystick.
  Joystick.begin(false);
}

// Helper variable for tracking whether the state has
// changed.
byte flag = 0;

void loop() {

  // Read the current joystick state.
  newUP = digitalRead(UP);
  newDOWN = digitalRead(DOWN);
  newLEFT = digitalRead(LEFT);
  newRIGHT = digitalRead(RIGHT);
  newBUTTON = digitalRead(BUTTON);

  // Check if the new state is the same as the previous
  // state. This way, we only need to send information on
  // the joystick state when it has changed.

  if (newUP != lastUP) {
    lastUP = newUP;
    flag = 1;
  }

  if (newDOWN != lastDOWN) {
    lastDOWN = newDOWN;
    flag = 1;
  }

  if (newLEFT != lastLEFT) {
    lastLEFT = newLEFT;
    flag = 1;
  }

  if (newRIGHT != lastRIGHT) {
    lastRIGHT = newRIGHT;
    flag = 1;
  }

  if (newBUTTON != lastBUTTON) {
    lastBUTTON = newBUTTON;
    flag = 1;
  }

  // If the flag is up, the state has changed and it
  // needs to be processed.
  if (flag) {

    // When building a new state, we start at zero.
    Joystick.setYAxis(0);
    Joystick.setXAxis(0);
    Joystick.setButton(0, 0);

    // Build a new state based on the newly read values.
    // The if clauses require an exclamation point to
    // invert the condition, since 0 = on and 1 = off.
    if (!newUP) {
      Joystick.setYAxis(-127); // up
    }
    if (!newDOWN) {
      Joystick.setYAxis(127); // down
    }
    if (!newLEFT) {
      Joystick.setXAxis(-127); // left
    }
    if (!newRIGHT) {
      Joystick.setXAxis(127); // right
    }
    if (!newBUTTON) {
      Joystick.setButton(0, 1); // fire button
    }

    // Everything is ready for transmitting the state
    // over USB. Before this point, nothing has been
    // transmitted yet.
    Joystick.sendState();

    // Remember to reset the flag once the new state has
    // been processed.
    flag = 0;

  }

  // Pause for one millisecond.
  delayMicroseconds(1000);

}
```

Program listing.



An ice fishing classic from Finland:

PRO PILKKI

Numerous bizarre and extremely Finnish videogames have been created in Finland over the past decades. And, if there is one game that is quintessentially Finnish, it must be Pro Pilkki.

Story by Jukka O. Kauppinen
Images by Mikko Happo, Jukka O. Kauppinen, Toni Kortelahti
Interview transcription by: Marko Koivuniemi, Antti Iiskola ja Juho Klapuri

Pro Pilkki, which came out in 1999, and its sequel have become legends in Finnish gaming history. This is quite an achievement for a game that was originally created exclusively for a Finnish audience, based on the authors' own passion and sense of humour. As chance may have it, however, Pro Pilkki is marching towards its 20th anniversary, more popular than ever.

As further proof of its significance, Pro Pilkki also received a place in the Finnish Museum of Games as one of the one hundred Finnish games from the last decade. Not bad!

Pro Pilkki games are made by Team Procyon, two gentlemen from Helsinki who are nowadays living in Kuopio and Oulu. **Mikko Happo** has studied biology, applied animal sciences and environmental health; in addition to his duties as a level designer, game designer and graphic artist, he has also modelled the fish in the game. **Janne Olkkonen** is responsible for the coding and additional design.

The duo visited Skrolli's bunker in Tampere in connection with the opening of the Finnish Museum of Games, and spoke at length about the history of Pro Pilkki.

Cannot get more Finnish

Ice fishers are a peculiar bunch of people. There they sit stubbornly, on the ice, braving the spring sun. Occasionally, they also end up sitting on ice floats and dipping into the water, as the approaching summer overcomes winter's might. The same type of bravery has also gone into making an ice fishing game, especially since you have had to spend countless hours on it, year after year, decade after decade. Perhaps it could be said that the game contains the same kind of creative madness as ice fishing, but this is hardly a surprise, since both authors are ice fishers themselves.

The guiltier of the two, however, might be Mikko Happo, since he taught ice fishing to Janne Olkkonen.

The duo now lives in different parts of Finland, but they used to be neighbours, within line of sight of each other. The boys met during the 1986 World Cup when the blokes from the neighbourhood got together on a nearby pitch.

– I had been fishing a bit with my father and grandparents, but Mikko suggested that we should go fishing on the Kajaaninjoki river. I had no proper equipment, so Mikko lent me his lures, and that was how it started, Olkkonen remembers.

Commodore 64 and computer games also appeared around that time, or shortly afterward.

– At first, we were simply playing all sorts of games. Programming started when they needed something. For example, when playing table ice hockey or table football, they noticed that a timer is needed. Well, a C64 can be used as a timer, so we simply studied how to do it, Janne explains.

At the same time, the duo learned many other things, both programming

and technology. Even the stopwatch was in development for a long time, until the MS-DOS era.

They also wrote their first ice fishing game at that time.

– I wrote the first ice fishing game, but it has been lost in the mists of time. It was a text-based game, and we did not really bear it in mind when writing Pro Pilkki. It might have existed as a concept in the background, but Pro Pilkki is not in the same continuum. At that time, we were more into doing level designs and using different game makers to learn about how games work. In addition to BASIC, we were also learning about making graphics, sound, music and other things, Mikko recalls.

The years rolled by and the brethren moved on to MS-DOS, but they kept experimenting with programming and different tools. Around 1994, they even wrote a fish diary application that sold 50–60 copies at a few dozen Finnish markkas each.

– We were mailing out disks as the orders came in. This was common practice at that time, Janne says. Mikko continues:

– Just two teenagers writing database software.

– Yes, but how many teenagers decide to write database software?, Janne remarks.

And that was not all!

Fishing in Kajaani

The unfinished, unpublished adventure game "Kalasta Kajaanissa" (Fishing in Kajaani) must also be included in the duo's fishing game series. Janne has the following to say about it:

– It was a poor man's point 'n' click adventure that took place around the river Kajaaninjoki. The story was quite good, actually. Sometimes, I feel like bringing it back to life and completing it; back then, I didn't have the skills. Well, in retrospect, maybe some of the ideas were a bit childish.

– It's the anarchy of youth. At this age, you no longer come up with that kind of stuff, Mikko quips.

The general feel of the adventure was borrowed from Monkey Island – where else? According to Mikko, the plotline made no sense at times, but in true Finnish fashion, all the characters had an annoyance meter that showed how ticked off they were. You could talk to them, slander them and tell them off. Depending on what you did, they would become more annoyed or calmer.

– Some actions were only possible when the characters were in a certain state of mind. By getting someone to flare up properly, you could do a specific thing.

The adventure also had different subgames, the first of which involved coin-tossing with the alkies on the shore. The goal was, of course, to steal the coins in order to buy bait.

The material for the adventure game still exists. We might see them at the Museum of Games one day.

And the next thing they did was something funny.

A fisherman inspired the fishing game

– You could create animations with Deluxe Paint, and Janne created a man catching perch. As we watched it and sped it up, we started laughing: this is like the guy from the local fishing shop in that his catch is so enormous. The salesman at the shop was known for being extremely skilled at ice fishing. Mikko resumes:

– We were watching the animation and started thinking about a game where you need to beat the salesman from the shop. Then we started making a plan: the lake could look like this, you could go there, what could you do and

What is Pro Pilkki?

Pro Pilkki is a Finnish indie ice fishing simulator published in 1999. Over the years, the free game has gained immense popularity and it is played by people of all ages and from all walks of life, in Finland and all over the world.

Briefly put, the game is about walking to a lake, contemplating where the fish might be hiding, and drilling a hole. Then, you sit down and wait for a nibble on your bait. The game is heavily based on genuine knowledge of fish and fishing and the actual science, which in part explains how it has become a lasting favourite among players and fishing enthusiasts alike.

What do Pro Pilkki and Monkey Island have in common?

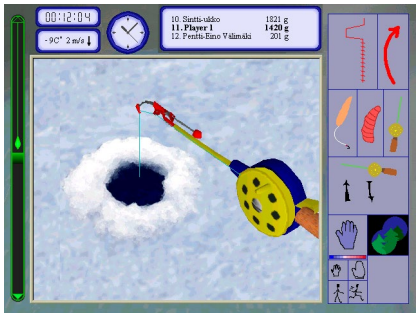
The first concepts of the 3D man walking on top of a background scanned from a wall calendar in early 2000.

The first prototype of an ice fisher from early 2001.

At first, the scenery was in 2D and the characters were 3-dimensional. Later, the decision was made to create the landscapes and trees in 3D as well. First screenshots. August 2001.

The first 3D rod, modelled after a photograph taken in 2001.

In 2003–2004, computer characters were added on the ice in order to model what the multiplayer competition looks like.



The user interface elements kept shifting around in early versions. The depth bar was a long time coming, but in this version it uses night colours which indicate that burbot is the desired catch. There are separate buttons for running and walking, and you can also see how cold your hands are. Even spying on your competitors has a separate button, inherited from Pro Pilkki 1.

The authors' ice fishing games

Commodore 64

Tekstipelipilkki

PC

Kalapäiväkirja

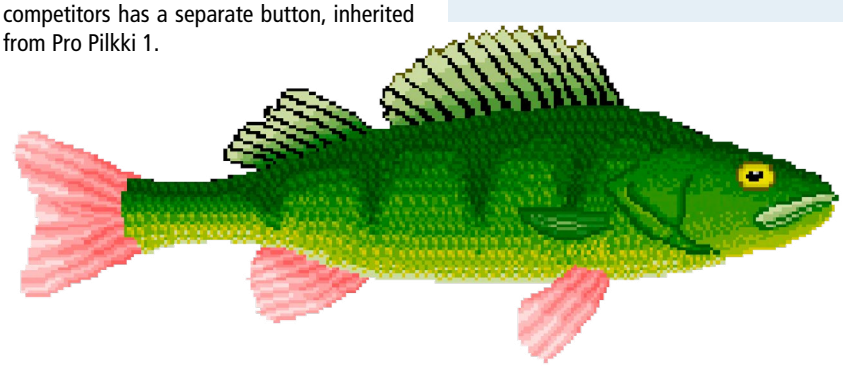
Kalasta Kajaanissa (julkaisematon)

Pro Pilkki (1999)

Pro Pilkki 2 (2013)

Android, Windows Phone

Pro Pilkki 2 Mobile (2015)



The first fish. Created in Deluxe Paint using the Pentium 200 MHz MMX PC that is now on display at the Finnish Museum of Games. The fish texture seen in an early graphics demo was soon replaced with a better version.

how could you catch the fish. This resulted in some very heated scribbling.

The animation became the salesman, the salesman inspired the idea and the idea became Pro Pilkki. Those with a keen eye can still find the legendary salesman in the game. He remains one tough fisherman. The journey from concept to game was rather short:

– It was only a year or two before we released 1.0, Janne remembers.

The duo started fishing by creating the tools. First, they created a lake editor and then the framework of the game. Most of the code was made in Turbo Pascal, but they also added assembly in order to make the game run quickly enough.

– We did not really publish anything before version 1.0, so the game was more or less done. One of the requirements for Pro Pilkki was that it had to fit on one floppy disk in order to make copying easier. The Internet was also gaining momentum at that time. I had a website since I was studying at the University of Oulu, and we used that to distribute the game.

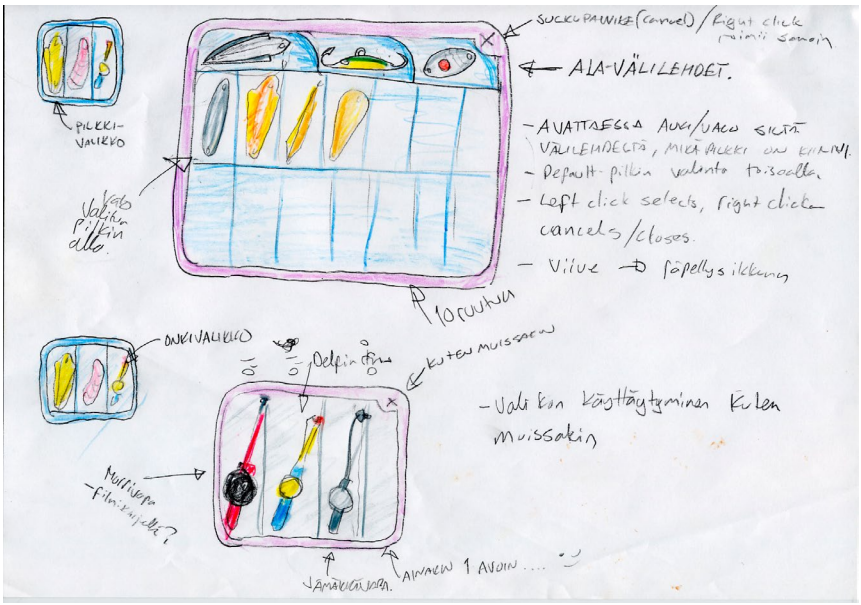
A fairly advanced feature was that the game had network play already at this point. Mikko recalls:

– We heard that people were playing the game on local area networks in computer classes. Then, the local ice fishing association in Joensuu contacted us and asked if we could have a Finnish Championship for Pro Pilkki. Why not, we thought. Once there, we noticed that the clocks on different computers were not synchronised, so someone caught all the fish before anyone else. Network play was not yet working properly.

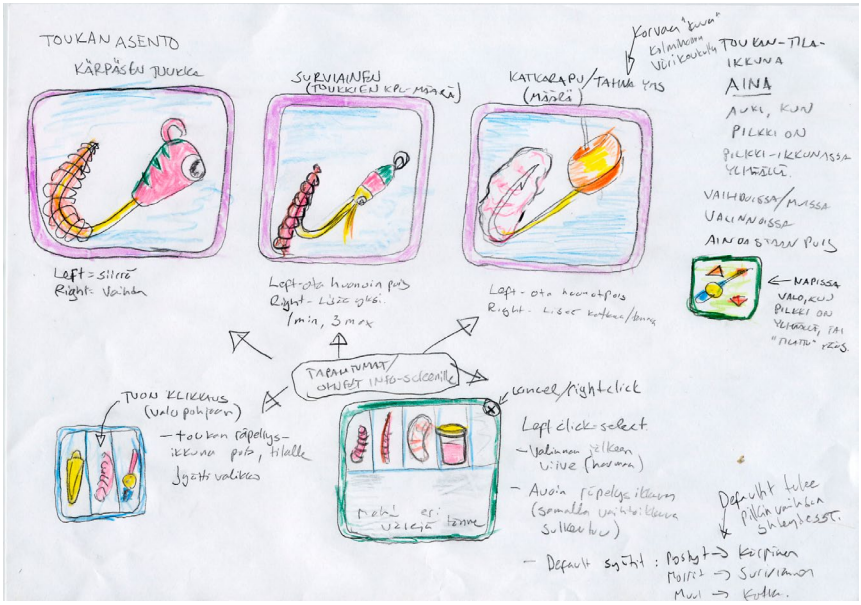
– It was nevertheless a step towards the current system, Janne says Mikko continues:

– Absolutely, we realised how much fun it was to play against a real person instead of a bot. It was easy to decide that Pro Pilkki 2 will focus heavily on multiplayer.

– The first version had a crude forest drawn in Paint and the base was made with the spray tool. After that, we thought about how to make the shorelines look prettier. One of the things we struggled with was whether to animate the fisherman. We were thinking about Theme Park and other contemporary games that had small Lemmings-like characters on the screen. However, this



A hand-drawn graphics and functionality concept draft for the larvae menu.



A hand-drawn graphics and functionality concept draft for the jig menu.

would have required much more animation, so we decided to use blobs. We did create an animation in the corner window, however, so that when you are spying on another player, you can see what they are doing: are the fish biting or not, Mikko recalls.

– And if you happened to be spying on another player right as their line broke, you could see swear symbols on their head. I don't know if anyone has ever seen them, Janne continues.

And this was not the only Easter egg. For example, there was a pike inside a well that you could only catch by drilling through a specific pixel. And someone actually tried that!

– One of the classic bugs was also that, once, two players caught the same pike at the same time, after which the counter multiplayer scoreboard wrapped around and suddenly there were 32,000 pikes there with their mouths open.

No room for chance

Instead of a random fish generator, the Pro Pilkki games are based on a full-scale scientific fish simulation. Mikko explains the background as follows:

– It makes no sense if the fish are random. You need to think about what ice fishing really is. There are days when you don't catch anything and other days when you do. The fish are in shoals or on their own. Their behaviour and appetite varies, the fish affect other fish, as do the fisher's own actions. We had to consider many different things.

Gaming for everyone

Pro Pilkkis are games that cross gender and age boundaries. Gamers and non-gamers alike can share the joy of fishing. Much like real ice fishing, actually.

– Ice fishing attracts young children and senior citizens alike, regardless of their gender. This also means that the audience for the game is very varied. The community for the ice fishing game also became quite unique. Many of the players don't know the first thing about technology, but if the real lake is not frozen, they need to use their computers, Mikko states.

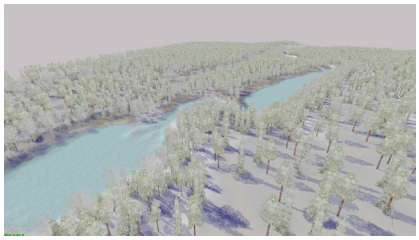
– Before Pro Pilkki, I don't think there was a fishing game with a community. And was there even a game with simultaneous competitive fishing? I think not. When the first game already had a multiplayer mode and the second version was built around it, there was time for a fantastic community and culture to develop.

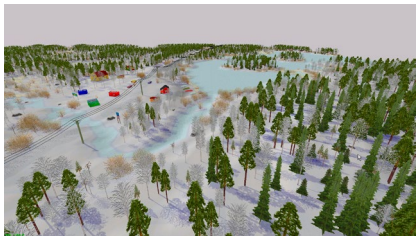
The postcards received by the authors also indicate its varied nature. The first game was received as "cardware", meaning that players were asked to send a postcard if they liked the game. The authors received hundreds of cards and small gifts, from very young players as well as grandmothers who had never touched a computer before but were now "sitting at the PC all day, ice fishing and swearing like a sailor".

Some of the cards are also on display next to the game at the Finnish Museum of Games.

Pro Pilkki is also an active LAN game and even has an eSports scene. Finnish championships are arranged for individuals and teams alike. The World Championship arranged in 2017 was the ninth consecutive event.

– What are the key questions? You have bait, different jigs and different rods. How you float the jig will affect the





Pro Pilkki 2 was practically a reboot of the game: it looked like a nicer version of the predecessor, but everything was new below the surface.

behaviour of the fish. We thought long and hard about how to convert these to ones and zeroes! I made a diagram where I explained what needs to be taken into account, how I think things are. After this, Janne took another notebook and started thinking about the mathematical functions in the background.

– After this, you sit on the ice and forget about everything beneath it. You no longer know what is going on below; you are on your own and need to think about where the fish are, why they're not biting here and where you should go next. Then, you need to look at the depth map and think.

– Around the release of Pro Pilkki 1, there were rumours of a cheat mode that can display the shoals of fish. We had to keep reassuring everyone that it does not exist.

Individual statistics and areas of occurrence have been defined for the different species of fish, for example. The general ice fishing simulation also takes into account the length of the day, which changes according to the season and also affects the appetite of the fish and the areas occupied by them. The shoal mechanics are complex; for example, the arrival of a pike in the area will scare away the small fish and decimate your catch.

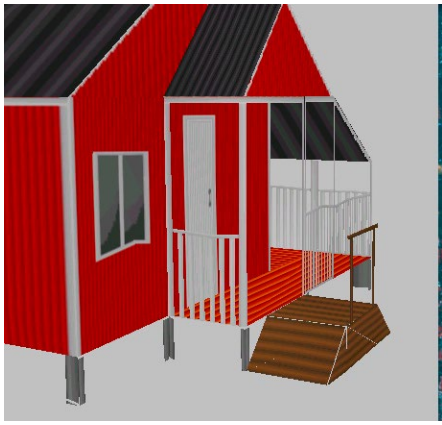
– Of course, we cannot reveal everything. There are many things hidden in there that players do not know. The lake editor has multiple levels that can be used to model these things and make them work together.

Still, this is a game. Realism is the goal, but the game also needs to be entertaining.

– However, fishers are a very demanding crowd. If the game is too unrealistic, they will disappear, Mikko says.

Of course, the game could still have many more things. Many features are the result of careful consideration, such as the number of mayflies. If you had unlimited bait, what would remain of the game? What, then, is at the heart of the game?

– Generally speaking, fishing games are based around tiring the fish, ensuring that the line does not break and that your catch does not stray too far. This mechanic is absent from the Pro Pilkki games. The game is not about tiring the fish; it is about knowing where and when to fish. If you want to catch something, you need to know what you are doing. Above all, Pro Pilkki is about tactics and strategy.



The buildings and items in the game were created in a dedicated object editor. A scanned film slide of the original building is also pictured.

Packing new bait

There is a time for everything. Pro Pilkki was a respectable 16 years old when it finally received an official sequel. Version 2 was mainly created due to the technical constraints caused by Turbo Pascal.

– It could no longer do what we wanted, so it was a natural transition, the duo says.

Pro Pilkki 2 was reprogrammed from scratch. The game now uses a 3D engine, the fish models are new, the online game was rewritten and automatically updated online scoreboards, new game modes and a host of other features were added. The most important target was to do away with the single-screen system and to make the map scroll together with the player.

However, once people are accustomed to something, they are very reluctant to give it up.

– Several players did not want to switch to version 2. For example, many

Fishing for jokes

Our discussion also took a few surprising turns. Such as the following:

JOK: They have the 24-hour race in Le Mans. How about Pro Pilkki 24? 24-hour real-time ice fishing!

Mikko: Pokémon style, with the entire map of Finland.

JOK: This could be the next version, ProPilkki Go.

Mikko: You go to the right place for ice fishing, but instead of a drill, you have a smartphone.

JOK: Drilling, fishing, staring at the screen.

Mikko: With a "beware of thin ice" warning in the loading screen.

were used to five-minute contests, when the shortest contest in Pro Pilkki 2 was 10 minutes. The culmination point was the Finnish Championship for Pro Pilkki, where we displayed version 0.1 or 0.2, the first demonstration version, Janne recalls.

– Even mothers with their children were playing, and we had one computer set up for Pro Pilkki 2. When an older couple sat there for what must have been over an hour, and the rest of the audience could only shoulder-surf, it reassured us that this version must also be quite addictive. They had specifically come to see Pro Pilkki 2 after hearing a rumour that it might be displayed here.

There was also some criticism; some people felt that version 2 was the same game with prettier graphics.

– Perhaps people did not see that we could no longer develop the first version.

Naturally, version 2 was made in a manner befitting the authors. The first thing to be created was the animation for the man drilling the ice, and it still remains in the game. Work on the lake editor only started after this. At this point, the game switched from 2D to 3D, which was a major leap for the authors, as well.

– I wrote the 3D engine for Pro Pilkki from scratch, mostly out of personal interest and to learn something new, Janne says and adds:

– The first part of the engine was my own object editor that Mikko used to create houses, trees and other things. I first tried Blender, but it was so rudimentary at that time that even I had trouble finding my way around it. I thought that there was no way Mikko would know how to use it, so I had to create one that he would agree to use for the graphics. Even so, he lost his nerve with it. At times, you had to compromise, with the houses in particular. But ice fishing was the main point, not the eaves on a building. Nowadays, we can import objects from Blender into the game, Mikko notes.

– First we made the object editor, then the lake editor and finally the game. The lake editor must have taken up half of the development time. Originally, the idea was to cut a few corners and use an online map to generate the game map. Well, that did not exactly work. If you were to put the fisher there in the correct scale, the transfer times could be up to several hours. At this point, we had to think about how much we need to reduce the scale in order to make the game fun to play, Mikko explains.

And the end result was good. Pro Pilkki updated the old game for a new age and created a good template for updates. Internationalisation has been strong and even the mobile version is doing well. A mobile version of the first game was also requested, but the code spaghetti did not allow it.

The story of Pro Pilkki is set to continue in the future, as the authors are still enthusiastic about it and have new ideas. The duo is not known for giving players any hints about future developments, but I was shown a beta build of the next version during the interview. The future looks bright – virtual ice will remain unaffected by global warming!

Not a two-man show

The authors wish to remind us that many other people have also participated in the making of the Pro Pilkki games. These include the following people:

Perttu Bergius maintained a list of records, manually tracking the fish caught by different players.

Jussi Mäntylä hosted the game's website – and later on the automatic list of records – on his Kalassa.net server.

Jari Ahola from the TTT-Theatre asked if he could do voice acting for the game together with a group of volunteers.

– We gave them a few pointers, but otherwise gave them free rein – and the results were great!

Veli-Matti Kananen offered music, since he had a background in music and wanted to work with video games. The quality of his offerings was so high that the authors discarded their own work.

Tero Konttila contacted the duo and showed them some of his work, asking if they wanted better menu graphics. Yes, they did.

Timo Korhonen has created textures for the fish, while **Anssi Nousiainen** worked on the fishers' overalls.

– We have received so much help for so many things. The best part about it has been that everyone has contacted us to ask if they could create something for the game, the authors happily explain.

Finnish madness for the world

While the Pro Pilkkis are clearly Finnish by nature, they have been surprisingly well received internationally, as well. Even though the first Pilkki was only available in Finnish, it received some international attention – and the language options in Pro Pilkki 2 made it much easier for the game to find new audiences. Surprisingly, some international players prefer Finnish since, according to them, the Finnish game sounds and speech samples feel more genuine.

Nowadays, Pilkki is played in more or less the same countries where ice hockey is a popular sport. The game is especially big in Russia, and keeps gaining popularity due to the mobile version in particular.

The growing international audience has also made some feature requests, since ice fishing cultures differ in different parts of the world.

– The Canadians have different species of fish, and they use tents and ice shanties when fishing. The Russians, on the other hand, wanted fish traps and flagline fishing. There are many different types of requests and ideas, but this is a Finnish ice fishing simulator where you follow the rules of Finnish ice fishing competitions, Mikko says about the international aspects.

Have you received any surprising messages or greetings from abroad?

– One time, we got a photo from South Africa that read "his first siika". A player from Ukraine sent us a picture of his friend, fishing on the ice. Apparently, the idea was to attract this friend to go fishing for real. And we even got a message from players in Brazil! These are the best greetings you can receive – pictures of people who have probably never been ice fishing, sitting at the computer and getting to know the hobby.

That time they nearly won a car

The "Revontulipilkki" is an interesting anecdote in the history of Pro Pilkki. I'll let the two friends explain it in their own words.

Holiday Centre Revontuli in Hankasalmi contacted us and said that they will be hosting a major ice fishing event known as Revontulipilkki. They asked if we could create an advertising version for the event and, young and enthusiastic as we were, we agreed. We created new menu graphics and five new lakes. One of them was the lake next to the hotel, where the contest was held.

We were also invited to the event for ice fishing. The first prize was the A series Mercedes, an even bigger model was offered in a prize draw, and outboard motors and scooters were also up for grabs. So we thought – let's go and collect the Mercedes!

The contest was peculiar in that the lowest allowed size was fairly high, I think it was 25 cm. The lake was full of perch, but they were smaller than that. Nobody caught anything, so we decided to try the spots that were good in the game.

We had consulted the local fishers for the game and created a map of the hot spots where fish could be found. After all, we couldn't create a random lake for the game. So we went to a place that was good for catching common whitefish in the game. We decided to let the perches lie – to win, we needed whitefish.

I had barely had time to drop my jig in when I caught one. It was not massive, but acceptable for the contest. I took it for weighing and the organisers announced that it was the biggest fish so far. After that, nobody caught anything for three hours. It started to dawn on me that I might actually win a Mercedes so easily. In the end, however, I [Janne] came second.

The winner got the Mercedes, but towards the end of the contest, he was in no condition to drive. I got to pick a prize and chose a scooter that I actually used to ride to work for a few years.

Ice fishing in a museum

Pro Pilkki is on display at the Finnish Museum of Games, complete with an author interview. The guys also donated their favourite jig and the computer that was used to create the first version back in the day.

Pro Pilkki is on display at the Finnish Museum of Games in Tampere, Finland – and there was never any doubt among the museum curators that it belongs there. Nevertheless, the authors were somewhat surprised at the interest. Mikko recalls:

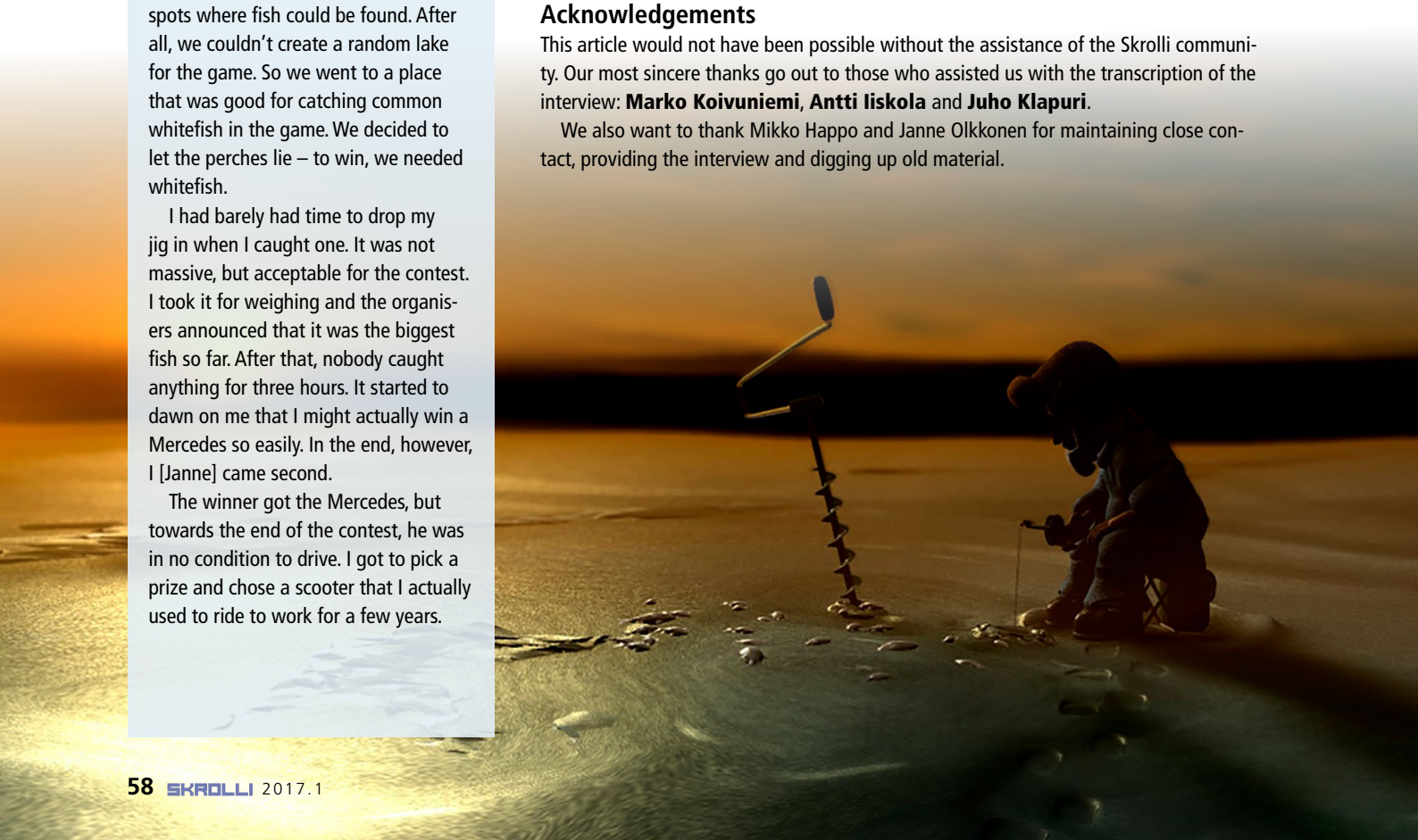
– When I saw the first news about the museum, I remember saying to Janne that I wish we could get Pilkki in there. Then the museum contacted us and we felt that this was somewhat easier than expected.

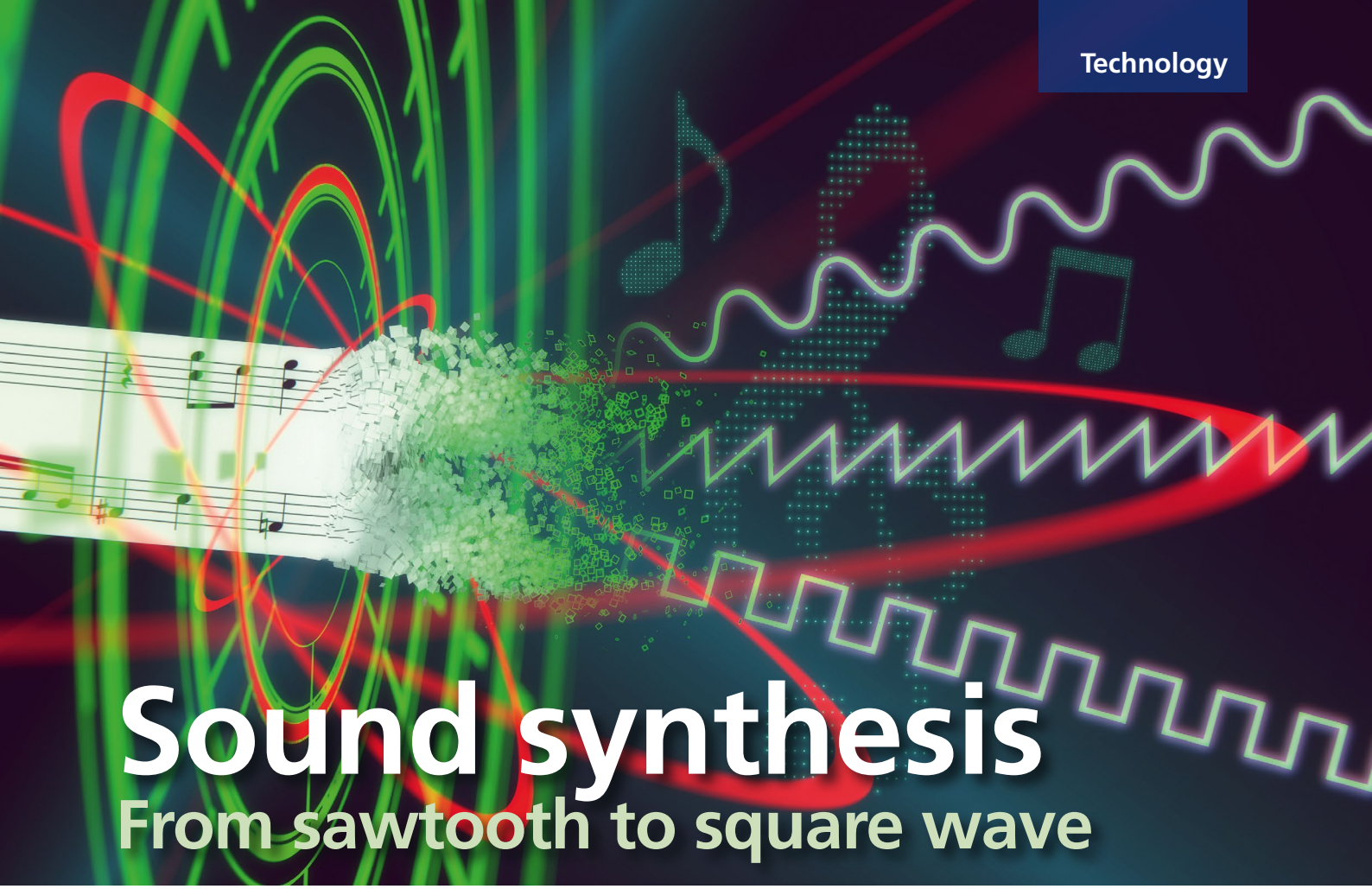
– After all, we created a game that had no precedent. Everyone who heard about the ice fishing game told us that you cannot make a game out of it. However, we showed that it was possible – and the game ended up in a museum. This was a wonderful reward for our work.

Acknowledgements

This article would not have been possible without the assistance of the Skrolli community. Our most sincere thanks go out to those who assisted us with the transcription of the interview: **Marko Koivuniemi**, **Antti Iiskola** and **Juho Klapuri**.

We also want to thank Mikko Happo and Janne Olkkonen for maintaining close contact, providing the interview and digging up old material.





Sound synthesis

From sawtooth to square wave

Computers can generate sound with a variety of techniques. What are the available options and could you write a soft synth yourself?

Story by Ville-Matias Heikkilä Images by Sakari Leppä, Chris Helenius, Mitol Meerna, Yrjö Fager

The short definition of sound is "audible vibrations in the air or another medium". Sound is generated in many ways: stringed instruments are based on the vibration of strung wires, human voice and wind instruments utilise sound waves bouncing from walls and speakers use an electromagnet to move a diaphragm.

Computers have also been making sounds since the beginning. Initially, the sound was the mechanical clicking of relays and punch cards, but the first speakers were connected to computers already in the early 1950s. A typical solution was to connect a speaker to the top bit of the accumulator register. This was useful for normal operation, since you could use the sound to determine whether the program was stuck in a loop, but it also made it possible to play music. By repeating a constant addition, you could flip the top bit from zero to one and back in standard time, creating a squeal at a standard frequency. The pitch could be altered by changing the constant.

The simplest sound devices in the microcomputer world, such as the beeper on the ZX Spectrum, more or less follow this principle: the speaker is controlled by a single I/O bit that can be changed in software. Of course, this takes up a lot of CPU time. The PC beeper is somewhat more sophisticated: it is controlled by a dedicated timer, so maintaining a steady beep does not require constant CPU intervention. However, the timer becomes useless if you want to play two notes at the same time on the PC beeper.

Both the original Spectrum and the IBM PC have fairly minimal sound capabilities compared to other machines of the era that have dedicated sound generation circuitry. A typical sound chip from the early 1980s has between two and four sound channels with adjustable pitch, waveform and volume. The SID chip on the Commodore 64 also offers programmable filters as well as the opportunity to use the sound channels to modulate each other.

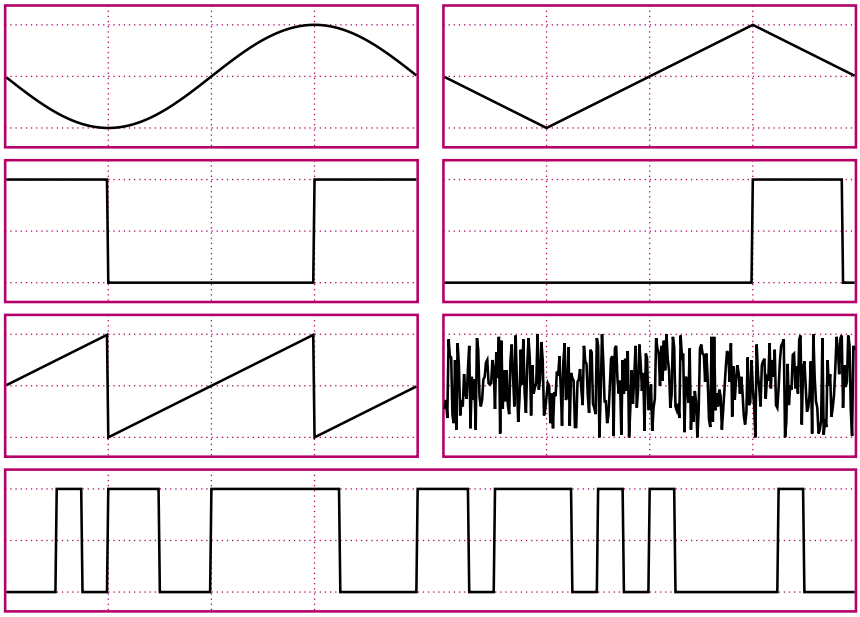
You can also improve the beeper by increasing the number of bits. The simplest possible "sound card" in the PC world is the Covox, which is practically an 8-bit D/A converter – a device that can produce 256 possible voltage levels instead of the two in the standard beeper. In theory, it is possible to play any sound by altering the output of the Covox thousands of times per second. The representation of sound as consecutive numbers that describe the variation of the voltage level is known as pulse code modulation (PCM).

The sound features in the Amiga's Paula chip are based on four 8-bit PCM channels that read the sound data from the provided memory range. In addition to the memory range, the playback speed and volume can also be adjusted per channel.

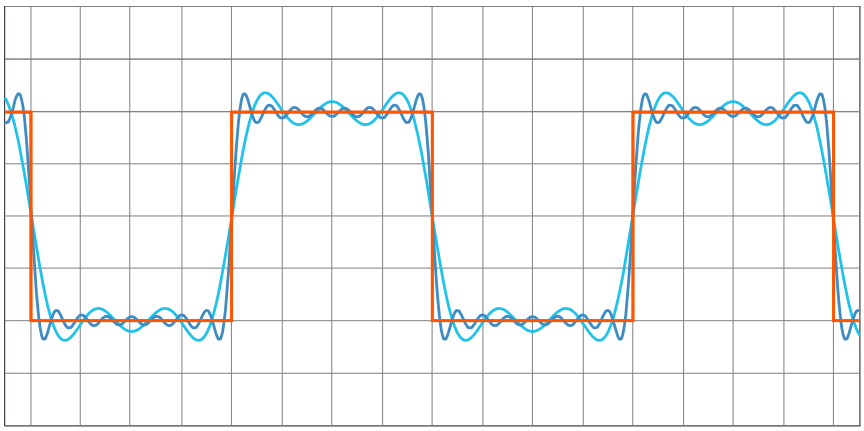
Nowadays, computers do not commonly generate music in real time – they simply decode it from media files, and the "most primitive" audio APIs do not even provide programmers with other opportunities. If however, the designer of the API has had some common sense, it also offers a PCM sound buffer that the program can fill with sound data generated through algorithmic processing.

Sound waves have shapes

An electrical circuit that produces a



Simple waveforms: sine, triangle, square, pulse at 22% ratio, sawtooth, white noise and an LFSR waveform used by Atari.



A square wave can be constructed from separate sine waves like this.

similar, repeating signal is called an *oscillator*. When this signal is turned into a sound, the voice is higher when the signal is repeated more often (frequency or wave length) and louder when the signal level variations are larger (amplitude). The timbre depends on the pattern formed by the variations, or the waveform.

A *square wave* is the simplest waveform in digital logic. It is generated when the signal varies between two extremes – ones and zeroes on a computer, for example. Nearly all digital sound chips can produce square waves, and this is the most "computer-like" or "eight-bit" waveform used in music.

A *pulse wave* is generated when the durations of two square wave phases differ from each other. Modifying the duration ratio through pulse width modulation (PWM) creates the gurgles and squeals commonly heard in SID music. Many contemporary digital devices use PWM to generate free-form audio signals: when a high-frequency pulse wave is routed through a low-pass filter, the differences in pulse width become differences in signal strength.

A *sine wave* is a fundamental waveform for both analogue oscillators and mathematical sound analysis. Any type of sound can be broken down into a sum of sine waves with different frequencies and volumes with what is known as the Fourier transformation. For the purposes of this operation, an individual sine wave represents a pure frequency – a sound that cannot be broken down into fundamental harmonics.

To the ear, a *triangle wave* sounds roughly the same as a sine wave, but it is easier to generate by means of digital logic. The signal strength varies in a linear manner from one extreme to another, creating an angular wave.

A *sawtooth wave* differs from the triangle wave in that it returns very sharply from top to bottom. It sounds much more pervasive than a square or triangle wave. It is also easy to produce in software by using a simple counter that wraps around.

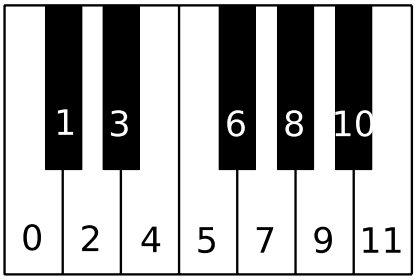
Noise is sound that lacks a regular sequence, or the sequence is so long that it cannot be heard as a pitch. The sound generated with a regular random number generator is known as white noise, an even mixture of all wavelengths – similarly to white natural light. If the noise is focused on lower frequencies and it reduces as the frequency increases, it may be known as brown, red or pink noise, for example.

Classic sound chips usually produce noise using linear feedback shift registers (LFSRs). LFSRs produce long series of ones and zeroes that sound different at low and high frequencies. LFSR chips can also produce shorter series that are commonly heard in the buzzing sounds of Atari's TIA and POKEY chips. The noise generator on the SID, for example, can also be tricked into generating shorter series with some precise timing.

The mathematics of sound

The basic waveforms presented above are by no means the only ones possible; they are merely among the simplest from a mathematical point of view. You can draw any type of curve for your waveform, and this is commonly done in chiptunes created with sound tracker software. However, all repeating waveforms have one common feature: when broken down into sine waves, they only contain wavelengths that fit exactly into their overall length.

If, for example, the frequency of the longest sine wave is 400 Hz, the sound



The Western scale with the ordinals describing the number of semitones.

may also contain frequencies of 800 Hz, 1,200 Hz, 1,600 Hz and so on. The sound represented by the longest sine wave is known as the fundamental tone and it corresponds with the pitch of the sound. Its multiples are known as harmonic tones, partial tones or overtones. The sound from natural instruments is not as clean. It also contains inharmonic tones that are not multiples of the fundamental frequency.

A significant portion of music is based on how the tones play in sequence and simultaneously. In order for different pitches to harmonise, they need to be within the correct mathematical proportions. If a pitch is twice as high as another one, their relationship or interval is known as an octave. A ratio of 3:2 is known as a fifth, and an interval close to a ratio of 4:3 is known as a fourth. These simple fractions allow the tones to match closely; one theory states that this is the basis for consonance.

In Western music, the octave is nearly always divided into twelve pitches, and instruments are most commonly tuned in a manner where a semitone, or the ratio between consecutive pitches, is a constant 2 to the 1/12th power. This creates a group of pitches whose relationships are fairly close to the fraction ratios. This type of tempered tuning is nearly exclusively used for computer music, although it is an unnecessary compromise by itself. Computers are not limited by the physical shortcomings of instruments, so it would be possible to recalculate the scale with every new chord, for example.

Many types of synthesis

When simple waveforms are added together, they play at the same time but otherwise remain unchanged. This is the basis for additive synthesis. In the computer world, it is mostly represented by "eight-bit" sound chips and chip music in general.

In the strictest and most theoretical sense, *additive synthesis* is exclusively limited to sine waves that can be used to construct any type of waveform or sound through addition. In its purest form, this was achieved in the ANS synthesizer built by the Russian engineer Jevgeny Murzin in the 1950s; it uses 144 sine waves of different frequencies to recreate a pattern drawn on a glass plate. Nowadays, reverse Fourier transformation is the most useful way of generating sound on the basis of a spectrogram created from scratch.

Another additive technique is known as *granular synthesis*, where "sound atoms" or extremely short samples or clips with standard waveforms are played as a condensed mass or "cloud" based on probability distribution, for example.

Subtractive synthesis may be considered to be the opposite of additive synthesis; it uses filters to dampen different frequency ranges. However, the term is not very precise, since resonance filters also amplify certain frequencies. Filters are very common in classic synthesizers, but appear fairly seldom in old sound chips – mostly because simple digital logic is not very well suited for them. The SID with its analogue filters is the major exception to the rule.

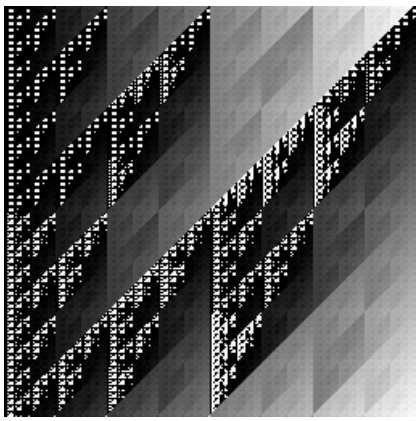
If the aim is to receive a realistic sound from a computer, using real-life sound samples is a simple and reliable solution. Tracker music, which started on the Amiga, is based on using samples as instruments. Sample music allows for using any kind of sound, but memory consumption has historically been a problem, especially since a realistic instrument sound requires more than simply playing back one sample at different speeds and volumes.

Synthesis techniques have been combined in many ways in order to receive all their benefits at the same time. One of these combination techniques is the Linear Arithmetic synthesis, used in Roland's synthesizers and PC accessories, which combines sampled elements with subtractive synthesis.

An entirely different starting point for synthesis is the modelling of the sound's physical creation mechanism instead of its frequency structure. There are many different modelling methods, but perhaps the most common is the digital waveguide that is well suited for mimicking string and wind instruments, for example. The models describe the progress of the sound wave in one-dimensional conduits that have interconnecting filters at different points.

```
main()
{
  int t=0;
  for(;;t++) putchar(t);
}
```

Listing 1. A C program that generates a sawtooth wave into stdout.

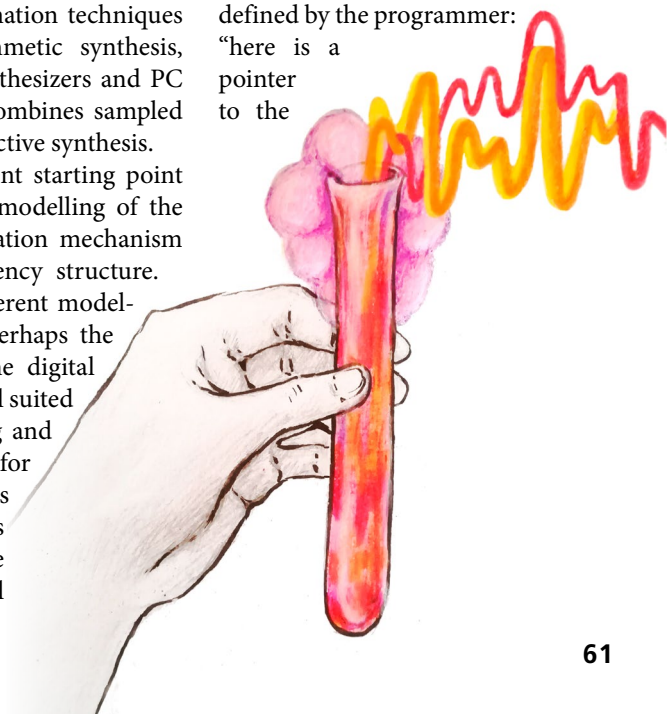


Bytebeat music often both sounds and looks interesting. The above graphics are generated by the expression `(t*((3+(1^t>>10&5))*(5+(3&t>>14))))>>(t>>8&3)`.

Filling the sound buffer

Perhaps the best way to learn about the basics of sound synthesis is to write your own software synthesizer or "soft synth".

The programmer of a soft synth usually starts with an API that accepts numeric values – to fill the PCM buffer. In the Unix world, music software may simply write bytes to standard output that is rerouted to the sound driver (/dev/dsp, aplay, pacat). Other APIs, such as SDL, invoke a callback function defined by the programmer: "here is a pointer to the



```
#include <stdio.h>
#include <math.h>

char notes[16*2] =
{
  26, 14,
  28, -1,
  29,  2,
  33, -1,
  38, 14,
  -1, -1,
  33,  2,
  -1, -1,
  29,  5,
  -1, -1,
  -1, 17,
  31, -1,
  28,  4,
  -1, -1,
  24, 16,
  -1, -1
};

struct
{
  float phase,freq,vol;
} chan[2];

int counter=0;

int next_sample()
{
  if((counter%6000)==0)
  {
    int row = (counter/6000)%16;
    for(int i=0;i<2;i++)
    {
      int n = notes[row*2+i];
      if(n<0)
        chan[i].vol/=2;
      else
      {
        chan[i].freq =
          pow(2.0,n/12.0)*100/48000.0;
        chan[i].vol = 0.5;
      }
    }
  }
  counter++;

  float out=0;
  for(int i=0;i<2;i++)
  {
    chan[i].phase =
      fmod(chan[i].phase+chan[i].
freq,1.0);
    out+=(chan[i].phase-.5)*chan[i].vol;
  }

  if(out<-1.0) out=-1.0;
  if(out>1.0) out=1.0;
  return (int)(out*32767);
}

int main()
{
  for(;;)
  {
    int a=next_sample();
    putchar(a&255);
    putchar(a>>8);
  }
}
```

Listing 2. A template for a simple software synthesizer that only uses square waves.

start of the buffer and its length, please fill it with sound".

Perhaps the simplest program that creates some sort of sound is an infinite loop that repeats a two-byte series. When the output is routed to a

```
INSTRUMENT NUM. 05  Snare
Attack/Decay    04  Vibrato Param   03
Sustain/Release E9  Vibrato Delay   00
Wavetable Pos   0A  HR/Gate Timer   02
Pulsetable Pos  09  1stFrame Wave   09
Filtertable Pos 06

WAVE TBL  PULSETBL  FILT.TBL  SPEEDTBL
06:FF 00  06:30 20  06:90 82  06:00 80
07:81 D8  07:30 E0  07:00 FF  07:00 20
08:41 00  08:FF 06  08:40 F8  08:02 60
09:FF 00  09:88 00  09:FF 00  09:02 00
0A:81 D8  0A:FF 00  0A:90 82  0A:00 18
0B:41 A8  0B:88 00  0B:00 FF  0B:00 10
0C:41 AC  0C:FF 05  0C:FF 01  0C:00 00
0D:80 D4  0D:82 00  0D:90 00  0D:00 00
0E:FF 00  0E:10 40  0E:FF 00  0E:00 00
```

Even advanced SID music trackers are heavy on the hexadecimals. The above defines a snare drum that contains one frame of noise (81) and two pulses (41), and the rest of the sound is dampening noise (80). Hex code FF 00 marks the end of a list.

sound driver accepting 8-bit monaural sound, the result is a high-pitched square wave squeal at a frequency equal to half of the sound device's sampling rate. Another simple program is an infinite loop that outputs the lower bits of the loop counter value, that is, an infinite sequence of the numbers 0 to 255. This creates a sawtooth wave at a frequency of one 256th of the sampling rate – for Unix sound interfaces, the default sampling rate is 8,000 Hz, resulting in a 31.25-Hz sound. Listing 1 contains the C source code for this.

By increasing the complexity of the putchar() function's parameter `t` we can create waveforms of different shapes at different pitches and volumes, and even music-like sounds with surprisingly little effort. One of the most concise examples is the expression `t&t>>8`, which modifies the square wave by switching its output bits on and off according to a counter that moves 256 times slower. Each output bit may be considered a square wave that the human ear can distinguish as a separate sound if it starts playing suddenly. If we add coefficients that are not powers of two, such as `(t*5&t>>7)|(t*3&t>>10)`, other intervals apart from octaves appear in the structure of the music.

These programs are referred to as bytebeats, and you can also find one on the Skrolli website at skrolli.fi/bytebeat/. The search for bytebeats was very active in 2011, and interesting expressions were discovered even through random experimentation. However, the technology was not more widely adapted even by the demoscene, where it might have offered an opportunity to create music in productions with only a few hundred bytes of data. Perhaps the waning popularity was due to the difficulties in controlling the technology and the general crudeness of the sounds.

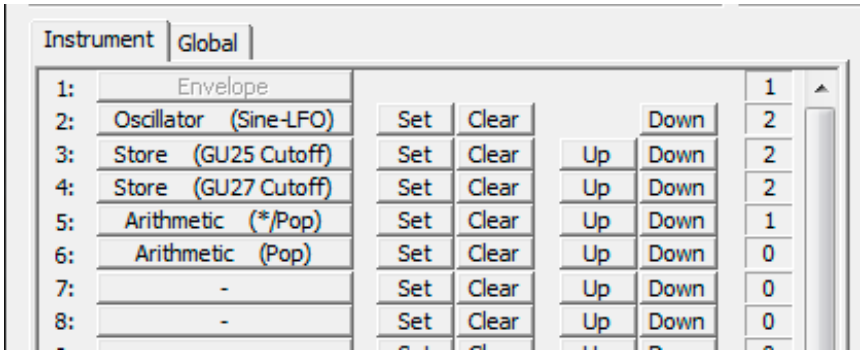
The traditional way

Regular soft synths differ substantially from the bytebeat programs. They are clearly divided into separate parts – for example, the low-level synthesis is separate from the "note reader". Instead of bizarre bit mixing and the exploitation of imprecise calculations, melodies and rhythms are stored in easily managed data structures.

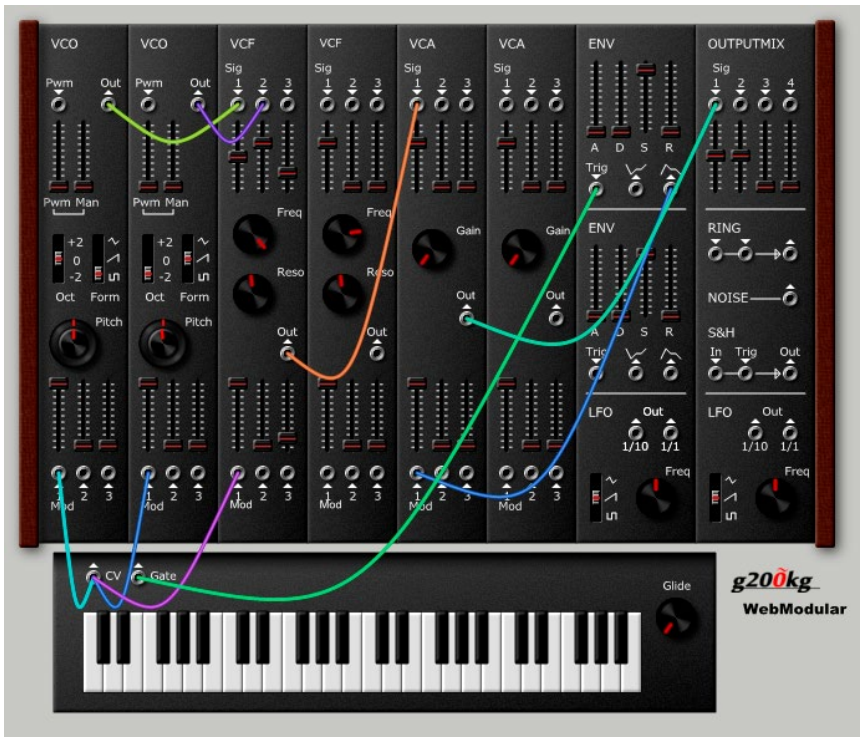
"Serious" composer software mimics notation in the manner for recording note data; separate duration has been defined for each note or pause. However, traditional demo music, tracker music and chiptunes use tables that run at a standard speed, and the duration of a note depends on how many blank rows are on the channel before the next note. This storage method is usually more beneficial for the computer, but it is difficult to use for describing more complex rhythms. However, for regular 4/4 pop music, this is not a problem.

Listing 2 describes a simple soft synth framework that includes a two-tone note pattern in a tracker style table. The table advances by one row every 6,000 samples, that is, 8 times per second with a playback rate of 48,000 hertz. As the table advances, a new frequency and volume are set for the channel. The frequency is calculated by means of involution according to the Western tempered tuning of 12 semitones; the number 0 in the note table refers to a basic frequency of 100 Hertz, 1 refers to an increase of one semitone and so forth. -1 is a pause symbol that halves the channel volume.

The example knows only one waveform – a sawtooth wave that is derived directly from the channel's phase counter value. However, it is deducted by half of the wave height in order to place it symmetrically around zero. The phase counter is kept between zero and one by using the modulus operation (`fmod`). You can change the waveform by modifying the contents of the line starting with `out+=` – for ex-



The instruments in 4klang are based on primary functions that are chained by using stack memory. The rightmost column shows how many values the stack contains after the operation.



The user interfaces of some soft synths based on block combinations also look like modular synthesizers. Pictured: WebModular.

ample, you can change it to a triangle wave at half the volume by using the absolute value function (`fabs`).

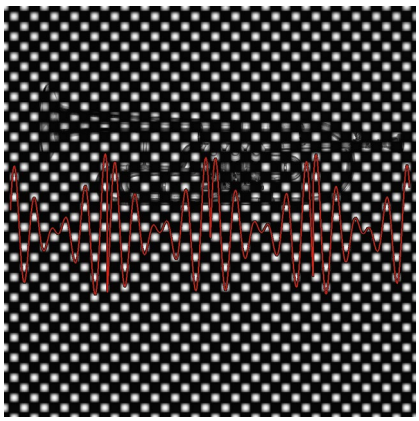
Classic home computers use sound chips that can individually synthesise sound and their operation is controlled by a player routine that is called at regular intervals. The player routine updates the registers on the sound chip, practically doing the same things as the example program does with the chan table when moving to the next note.

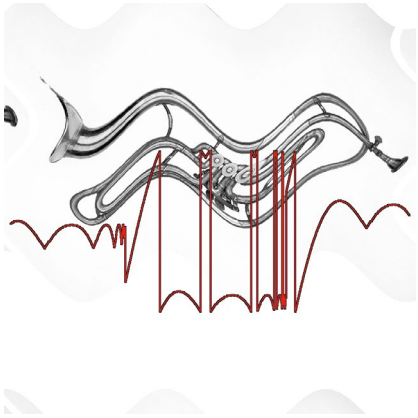
In SID music, for example, the player routine is usually called 50 times per second or at every PAL screen refresh, and it does not move to the next note every time. Instead, it can vary the sound parameters and create dynamic short-term effects. A typical SID drum sound, for example, varies between noise and pulse waves at different frequencies. Furthermore, you can play chords on a single channel by repeating the notes in a loop, one after the other (arpeggio).

In addition to instrument-specific command sets, the player routine can also perform different effects, such as gradually sliding the pitch between notes. These types of effects are especially popular in sample based tracker music.

More dynamics

Of course, a soft synth does not need to be limited to simple waveforms or the traditions of chiptunes or tracker music. The real fun begins when you start tweaking the synthesis algorithm.



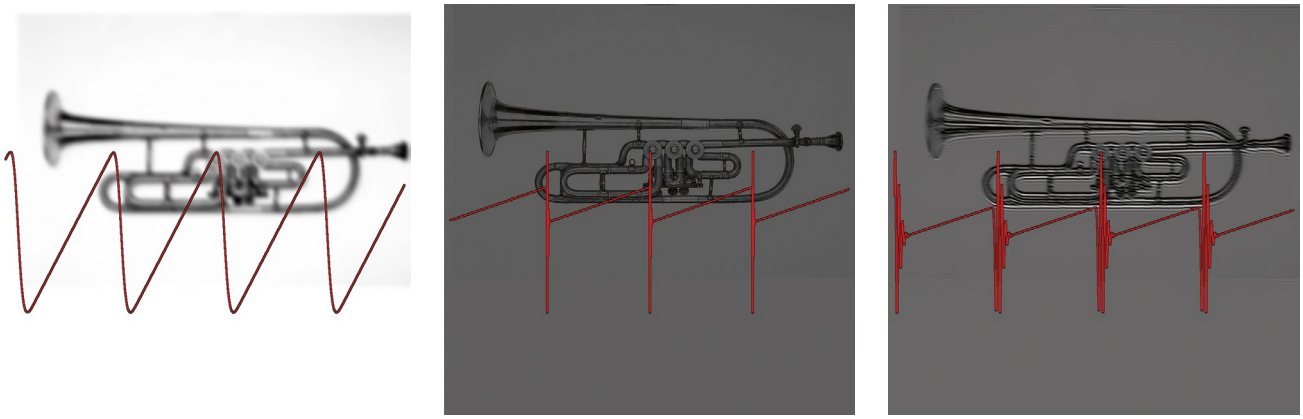


This is how a sawtooth wave is amplitude and frequency modulated when controlled by a sine wave. The trumpet images demonstrate what the same algorithms do to graphics.

A simple waveform oscillator usually has two parameters – frequency and volume. If these remain constant throughout the duration of the note, the result will be a somewhat mechanical beep. However, if we modulate these parameters by employing a secondary oscillator, for example, the sound becomes much more natural.

By adjusting the volume with an envelope, you can make it sound like the volume variations on a natural instrument, for example. Envelopes typically use four parameters: attack (the rate for moving from silence to maximum volume), decay (the rate for returning to basic volume), sustain (basic volume) and release (the rate for reducing the volume back to zero). The envelope usually remains at a basic volume for as long as the "key on" signal is received, after which it moves to the release phase. Naturally, the envelope may be substantially more complex than the basic ADSR model, as some of the late 1990s PC trackers demonstrate.

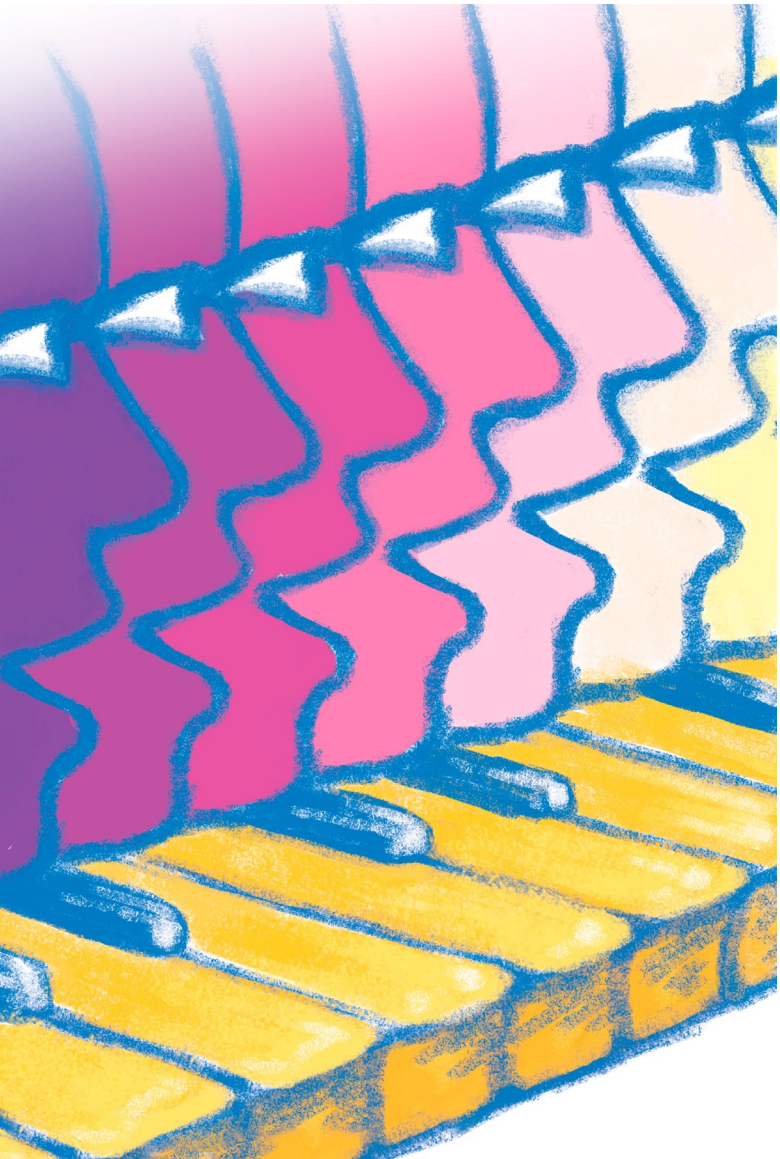
When frequency or volume is modulated with a constant waveform, such as a sine wave, the result is a vibrato or



The effects of low pass, high pass and resonance filters on a sawtooth wave and the picture of a trumpet.

tremolo sound. If the modulator frequency is at least equal to the carrier wave, this results in frequency or amplitude modulation (FM or AM). Amplitude modulation is also commonly known as ring modulation due to the shape of the analogue circuit that implements it. Frequency modulation, in particular, creates many inharmonic tones that do not occur in either of the original waveforms. In the computer world, FM synthesis has been used in the OPL chips found on AdLib and SoundBlaster cards, which may have given it an undeservedly bad reputation among computer hobbyists.

Of course, many other parameters can also be modulated by means of envelopes and oscillators. Some examples include the pulse waveform width ratio, the filter threshold frequency and the mixing ratio for two waveforms. If the sound generator is not an ordinary waveform oscillator, its adjustable parameters may be completely different. Of course, you can also use the results from other modulations as the source of the modulation – there are nearly endless possibilities for combination.



You can experiment with combining different synthesis functions by using traditional programming languages as well as bespoke sound synthesis development environments such as *Supercollider*. Many applications allow for combining blocks graphically in either an abstract diagram view or a user interface that is reminiscent of a real modular synthesizer.

The block network allows for creating a fairly versatile selection of sounds compared to the small amount of data required for describing one. This has made it an attractive approach for music routines in small scene demos. For example, the instruments in the 4klang soft synth, designed for Windows based 4k intros, are based on structures described in a simple stack-based language.

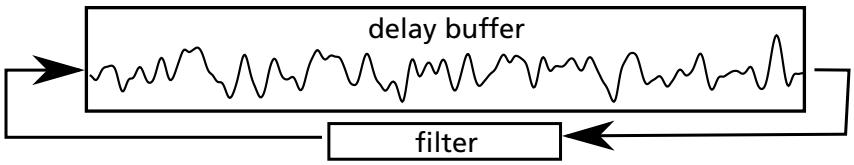
Filtered sound

The different synthesis techniques often leave high-pitched squeals and other unwanted frequencies in the sound. In this case, using filters to smooth out the frequency composition is a good idea even when you do not want to use them for added effect. Of course, filters are also useful in the synthesis itself, since you will often want to change the frequency composition of the sound over time.

A *low-pass filter* dampens higher frequencies but allows lower frequencies through. In programming, you can implement it by mixing some of the results from the previous filter operation into the sample – or, to make it really easy, simply by calculating the average of the two. This corresponds to the blur filter in image processing.

A *high-pass filter* is the exact opposite; it leaves the portion that a low-pass filter would remove. The principle for implementation is the same, but the mixing factor is set to negative. The high-pass filter corresponds to the sharpen filter in image processing.

A *resonance filter* is especially interesting in terms of sound synthesis,



The operating principle of the Karplus-Strong algorithm.

since it amplifies specific frequency ranges. Thus, it can be used to generate formants, harmonics augmented by resonance. When implemented digitally, a resonance filter is somewhat more complicated than the above two – it calculates a weighted average of the sample and the two previous filter results.

The phoneme distinctions in spoken language are largely based on the differences between formant frequencies. Therefore, *formant synthesis* that uses filters is a fairly popular method for speech synthesis. In this model, the "throat" generates either noise or a gurgle similar to a dampened sawtooth wave, depending on whether the phoneme is unvoiced or voiced, and this "throat wave" is modified with a series of resonance filters. Three filters are sufficient for generating all the vowels in English, but a more complete model of the vocal tract requires a few more.

Bouncing sound waves

Adding echo to the sound is best done with a delay buffer that supplements the sound with its own history from one second ago, for example. Different types of spaces can be simulated both by altering the duration and intensity of the echo and by filtering the echoes and placing several echoes on top of each other. Naturally, having multiple speakers and calculating the bouncing of the sound separately for each of them will intensify the effect. *OpenAL* is an interface for producing positional sound that has been especially used in games.

When wanting to model instrument acoustics instead of echoing spaces, one opportunity is to use shorter delay buffers with lengths equal to the wave lengths of the desired sounds.

Perhaps the simplest possible physical instrument model can be built with the Karplus-Strong algorithm. It first fills the buffer with noise and then plays back the contents in a repeated loop while applying a low-pass filter. This creates a surprisingly authentic guitar sound. In this model, the buffer corresponds to a string along which the wave advances, finally meeting its point of attachment from where it bounces in the opposite direction, slightly filtered. Since the advancement is similar in both directions, they do not need to be separately modelled.

The pitch of the sound in the Karplus-Strong model can be altered by varying the length of the buffer, and the varying of the filtration function also affects the damping rate of the sound, among other things. By creating distortion and feeding it back as string vibration, you can simulate an electric guitar. Negative filter parameters turn the sound into metallic pings.

More generic versions of Karplus-Strong are known as *digital waveguide* synthesis: there may be several different filters and links along the buffer and there may even be several buffers. In addition to string instruments, they are especially well suited for modelling wind instruments. Waveguide models can also be extrapolated into two or more dimensions, which allows for the simulation of drum skin vibration, for example.

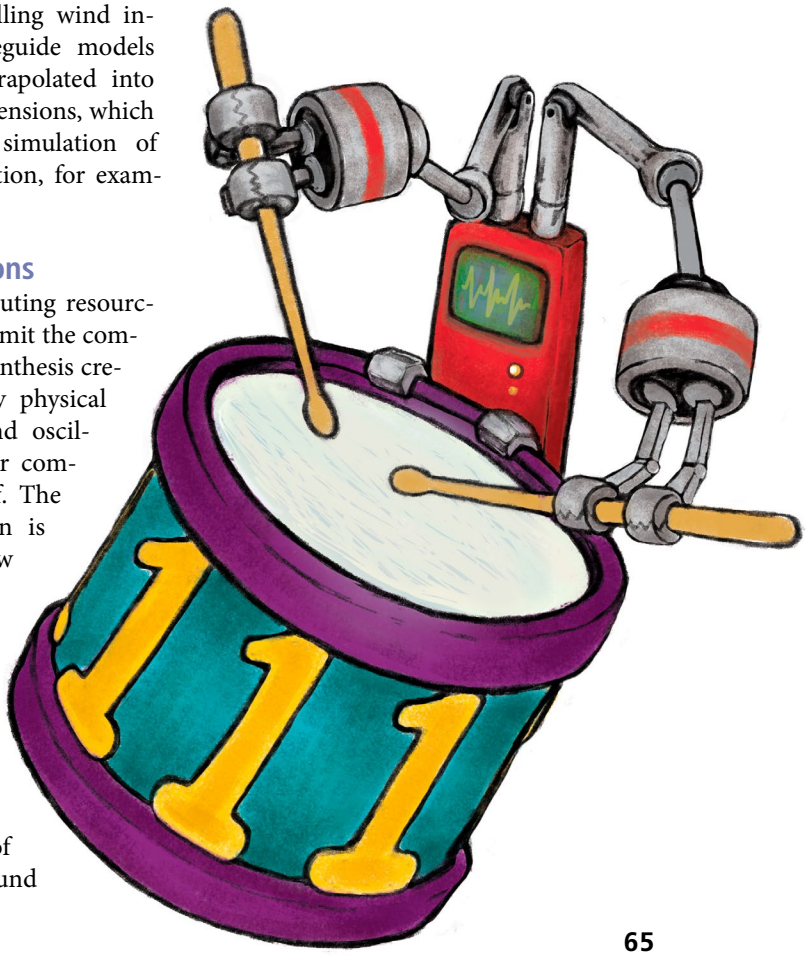
New dimensions

Nowadays, computing resources do not really limit the complexity of your synthesis creations – be they physical models, filter and oscillator networks or combinations thereof. The essential question is how to find new and interesting opportunities among these offerings.

Neural networks have been successfully tested in many areas of music and sound creation. Recurrent networks can be used as synthesizers by themselves: the neuron weights are like knobs that the learning algorithm rotates to match the provided sound template, for example. Of course, the network can also adjust the parameters of an existing synthesizer and, for example, learn to play an imaginary instrument that is too complex for a human.

The next revolution in sound synthesis may not require a massive amount of computing power. There are most certainly lightweight synthesis models that no one has yet thought about. A few years ago, the bytebeat experiments revealed that even the basic computing operations may still be used in pioneering ways for sound synthesis, even though the hardware required for them has existed for decades.

The purpose of this article was to provide a superficial basic understanding of sound synthesis and the related programming techniques. There is a vast number of programming languages, APIs and sandbox environments available for your experimentation, so simply find one you like and start building sounds!



FAIL!

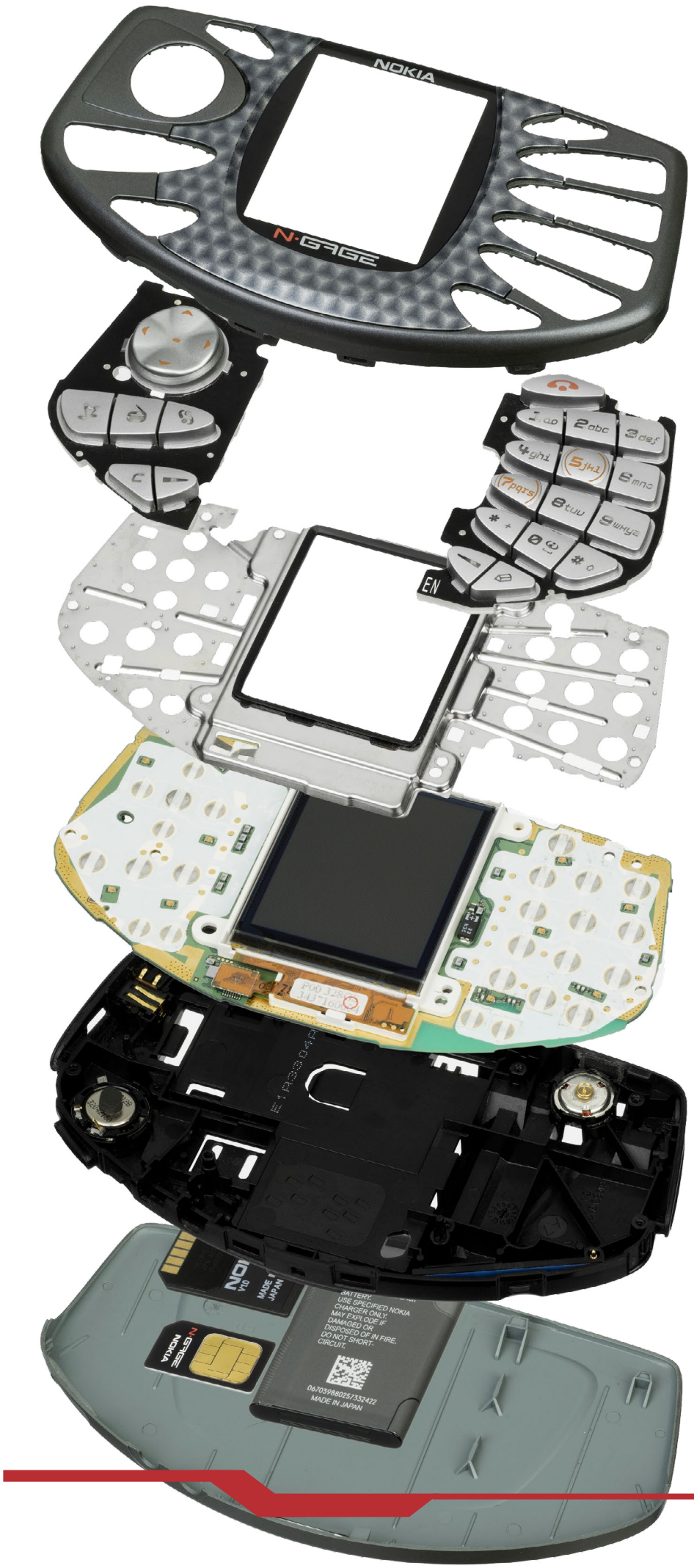
Just a side note

At its best, Nokia was a company that could do anything. However, it was completely clueless about making a game console.

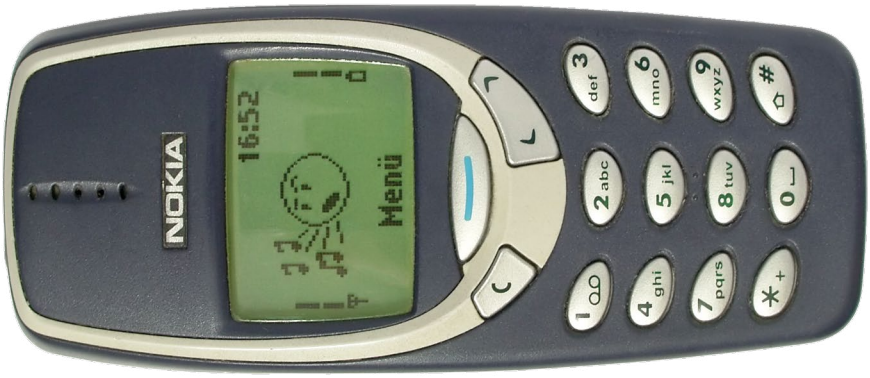
Story by Mikko Heinonen
Pictures by Marko Haarni, Wikimedia Commons users Evan-Amos, Shritwod, Rainer Knäpper

Nokia mobile phones were a unique stroke of luck for Finland. With the proliferation of GSM, a small country recovering from an economic recession gave rise to a company that soon put its products in everyone's pocket. Even though Nokia Mobile Phones the company no longer exists in that same form, its best handsets are still remembered as synonyms of durability and easy usability.

The consumer phones were selling like hot cakes, but Nokia also had its own skunkworks projects. You may recall the lipstick-shaped 7280, the



3650 with its round keypad and the N90 which turned into a video camera. Most of these curiosities were never meant to conquer the world. Instead, they were used to generate interest at industry events and to show that the giant could still innovate.



Nokia 3310.

A game console by chance

One such curiosity was the Nokia 5510, introduced in 2001. It was a development of the popular 3310 that featured a QWERTY keyboard and a music player that used MMC memory cards. It was not a massive success, but nevertheless sold well enough to warrant the design of a successor. Nokia was shifting to colour displays, so the new music phone was also getting one. The miniature full keyboard was axed in favour of a directional pad and phone keypad.

With these updates, the new Symbian Series 60 handset started to look like the Nintendo Game Boy Advance. However, it remains slightly unclear at which point the decision was made to turn it into a game console; according to some ex-Nokians, the idea only came about when the hardware was very nearly complete. Even then, the designers were told to use as many standard parts as possible. Nevertheless, in 2002, Nokia announced that it was going to introduce a new competitor to the handheld video game console market that was dominated by Nintendo.

Grab the taco and sidetalk!

The gaming phone was called the N-Gage, and it was packed to the brim with state-of-the-art technology. Of course, the Nokia family heritage made it a versatile phone, but using Bluetooth or the Internet for multiplayer instead of separate link cables had not been seen before in this type of device. Even though the screen was small, it was fairly good and had a backlight, whereas Game Boy Advance players needed to carry an artificial sun with them. The device even contained an FM radio.

Unfortunately, the original N-Gage also contained errors in design which indicated that the entire game console side had been an afterthought. Perhaps the biggest problem was that switching games was a chore: in addition to switching off the power, you needed to remove the battery, since the memory card slot was below it. This was not a problem for phones where memory cards only act as extensions of storage space. In a gaming device that doubled as a phone, this was a real deal-breaker for many.

And the worst was yet to come. Fitting all of this technology in a tiny shell meant that Nokia had to compromise on the phone itself. The speaker and microphone were moved to the top edge of the device. When calling, you had to hold the N-Gage in a peculiar position next to your head. To make matters worse, the design was reminiscent of a taco shell. The Internet went crazy and "sidetalking" became a viral phenomenon: people were taking countless photos with crazy things sideways next to their ear. N-Gage was now officially a joke, and this type of publicity is difficult to monetise.

No help from games

History has shown that, when treated to a steady stream of killer apps, the general public can be very forgiving. Unfortunately, N-Gage was further hurt by Nokia's inexperience in the game industry. The system launched with big names like Pandemonium, Puzzle Bobble, Sonic the Hedgehog and even Tomb Raider, but there was no real reason to buy this system over any other.

And since the N-Gage was basically a mobile phone, the ports of AAA titles from home consoles were mostly lacklustre. The washed-out colours of the small screen made puzzle games like Puzzle Bobble and Puyo Puyo difficult to play, as players had a hard time distinguishing between the different blobs. It also did not help that the aspect ratio was obscure: while 11:13 might have been optimal for phone use, players familiar with 4:3 found it bi-

Nokia 5510.

Nokia N-Gage QD.

zarre. Many of the games came from platforms with 3D acceleration, but the N-Gage could only do software rendering. The end result was most often slow and unattractive.

Another problem was that the plan was to sell the device at phone shops, as was natural for a Nokia device. However, few of them were willing to stock large amounts of game memory cards for a single phone model. Meanwhile, game stores were not a common place to buy mobile phones, so Nokia had a bit of a catch-22 on their hands. Offering games for download online might have helped, but unfortunately this was beyond the scope of Nokia's innovations.

Quit the Discounts

There was no way Nokia could have been happy with the N-Gage's performance after a few months. Sales were slow nearly everywhere, and large retail chains were soon offering heavy discounts. In the United States, the Game Boy Advance was outselling the N-Gage at a rate of 100 to 1. They had two options: give up or try to fix the product.

The answer was the N-Gage QD. It was a better proportioned and more ergonomic device that arrived only six months after the original N-Gage and seemed to solve many of its problems: there was no more sidetalking and you could switch games without turning the system off. However, the FM radio was gone along with a few other accessories, and the screen resolution and aspect ratio stayed the same for legacy reasons. Nokia did find a better panel for it, however.

The company also started focusing on game development. For example, Nokia contracted RedLynx to write the excellent Pathway to Glory and High Seize; both games were specifically designed with the N-Gage in mind, not merely ported from another system. In fact, the only problem with these games was that they came out in late 2004 and early 2005, by which time the milk had very much been spilt already. Perhaps their biggest contribution to history was that they helped fund a studio that went on to do great things.

While the QD was by no means faultless and definitely not a very powerful game system, it was a much better effort than its predecessor. If it had been available immediately, Nokia would have been spared the embarrassment of sidetalking and the universal disgust for the difficult game switching.

Life after death

At the end of 2005, Nokia finally admitted what everyone else already knew: the N-Gage was a failure and sales were only a fraction of the targets. Approximately three million devices were sold overall, while the goal was to sell four million by the end of 2004 alone. Out of these, many were sold to customers who noticed that the smartphone features in fact made the QD a very capable mobile device, even if you totally forgot about the games.

The N-Gage had one more coming when Nokia relaunched it in 2008. This time, the name was given to an online gaming service that ran on Symbian S60 and was slightly reminiscent of Xbox Live. It had a lukewarm reception, not least because Nokia was losing its gamer customers to Apple. The new N-Gage was shut down in 2009.

No hard feelings

Over the years, the N-Gage has been the laughing stock of many. Despite all this, it remains the only multi-million selling game console designed in Finland. The money Nokia poured into N-Gage development also boosted the Finnish game industry at a very suitable time.

So far, no one has been able to touch Nintendo on the handheld console market – and it seems that this is the final score, since the Switch is no longer purely a handheld system. Mobile gaming now mainly takes place on smartphones – which is exactly what Nokia tried to do 15 years ago. Even contemporary reviews stated that the N-Gage was a good idea that was poorly executed.

At that time, the sad fate of the N-Gage did not affect the mammoth Nokia in any way. In what could almost be considered an act of defiance, it launched another successor for the Nokia 5510: The Nokia 3300, which looked exactly like a game system but wasn't one. Its US model even retained the 5510's full keyboard.

Nokia 3300.

Virtual nightmares

Why is virtual reality still so scary?

Janne Sirén

The Oculus Store ranks VR contents on a three-step scale, from comfortable to intense. The primary purpose is most likely to warn about the adverse health effects of VR, since some people suffer from motion sickness-like symptoms while using it. Secondly, this is a reference to the intenseness of the effects.

And this is the greatest challenge for a VR beginner. You see, VR can be unbelievably scary.

Personal space

The first time that I was startled by VR was when I was looking at a 360 degree beach shot on the Gear VR. As I turned my head, the sea, rocks and distant boulevards gave way to a person standing right next to me. This was only a static photograph, but as the headset fools your brain into thinking everything is in real size, a nearby person – even a harmless tourist frozen in time – was a surprise.

Sound and movement add to the intensiveness. The HTC Vive tutorial includes a futuristic living room where you test the operation of the headset and the sound through the speakers. Amazement turned into terror, as the floor suddenly opened up with a loud rumble as I was gazing around the room.

The Last Sniper VR, set in World War II, starts with a scene where you parachute out of an aircraft. I had the same feeling of discomfort as on the "beach" when I had other soldiers waiting to jump standing in front of my face. Personal space is a big thing in Finland – we are famous for standing metres apart from strangers on bus stops – but I would expect this virtual proximity to feel intense anywhere in the world.

The virtual parachuting was not uncomfortable to me (quite the opposite, actually), but some users with vertigo have requested an opportunity to skip it. However, even I had to take the headset off during the first German land offensive.

They say that horror movies are the scarier the less they show. Sometimes, even the notion that something scary might happen becomes scary. In the *Rilix VR* rollercoaster simulator, I was constantly on edge about when something will attack me and fill my entire field of vision.

I cannot exactly remember the last time a computer game got me so scared. Perhaps it was *Friday the 13th* on the Commodore 64 in 1986 – the game would occasionally flash images of murder to the player. Even though the game was running on a small portable TV, the unpredictability was scary to a 10-year-old. In VR, the program code can take over your entire vision and hearing, which can be scary even for an adult.

Find your inner coward

So, has VR revealed that I am actually a coward? Undoubtedly, parachuting over Normandy in June 1944 would have been a scary experience. Nevertheless, VR fear is different. You may feel dizzy during a virtual experience, but the headset cannot convey a fear of death or the sensation of falling.

Virtual reality is scary because it takes over your senses in a holistic but deficient manner. A normal person's field of view, approximately 180 degrees, is larger than the 110 degrees of the current VR headsets. In the real world, what you see in the corner of your eye can prepare you for turning your gaze. In traditional video games, the limited field of view is safely boxed in on the screen. In VR, however, turning your head will always reveal a surprise.

Virtual reality is also scary because instinctive reactions are not always met with the expected response. Depending on the limits of the virtual system and the implementation of the software, the VR response will always be more of a close approximation. This does not occur in the real world, where the laws of physics apply. Even the admittedly phenomenal body and hand tracking of the HTC Vive inside a limited space is not similar to handling a real rifle, let alone moving around with it.

However, the largest problem is that we sense our environment in countless ways. In the real world, we observe things happening next to us before we turn to look at them. We can hear the slightest crack, feel the heat and movement in the air, our body senses things and we can see in the corners of our eyes. Many of these cues are missing in virtual reality. In the real world, it is also much harder to teleport someone right next to us or to arbitrarily drop the floor below us.

Virtual reality is like a dream. It can be a sweet dream or a nightmare; everything seems familiar, but nothing quite works as you would expect. Waking up, or returning to the real world, might be a relief – at last, the world is working as it should.

Nevertheless, I wouldn't give up dreaming.


SIMULAATIO 7
DEMOPARTY 20.-22.5.2016 // JOROINEN // FINLAND
simulaatio.org
LOADING
API
KEWL

Simulaatio
– The last real demo party in Finland

What is the demoscene about? How does it feel to be at Finland's "last real demo party" at Hotel Joronjälki in Joroinen?

Story by Valhe Kouneli Photos by Antti Kiuru, Terho Tanskanen, Matti Hämäläinen

Demos are an abstract, computer generated real-time art form. The demoscene subculture is less known in the United States, but in Europe and the Nordic countries, in particular, it is still going strong. Its roots are in the pirated games of the late 1970s, when cracker groups would add their own introductions, known as "crack intros", or cracktros for short, to any games they released. Eventually, the cracktros were taken out of their original context and the gorgeous, often music-synchronised visuals became known as demos and intros.

Demo parties are events where people gather to compete for the title of best demo. There may be several categories: best demo for the Amiga, best four-kilobyte intro, best demo without platform and size limitations. At the largest parties, the winning groups may receive thousands of euros, but most often, fame and glory are the main rewards.

The demo parties have given rise to a specific culture that combines technical expertise with overall creative insanity and all possible related phenomena.

Friday, 20 May 2016

The coach from Helsinki to Joroinen, a small town of 5,000, leaves at around five in the afternoon, carrying a group of loud demosceners already in a very festive mood. Another similar coach passes through two other larger cities, Tampere and Jyväskylä.

The Simulaatio party has been arranged six times, but this is only the second party arranged at Hotel Joronjälki, a notorious hotel of horrors that appeared in the Finnish version of Hotel Hell. Last year's party was a success, and this year's accommodation was also sold out instantly. A couple of free rooms became available at the last moment, so here I am, sitting in the coach from Helsinki.

We will soon leave the capital behind us, and **vELiKANi** (everyone here goes by their scene alias) uses the PA system on the coach to welcome everyone. The green welcome jello shots are distributed in cardboard cups. The party has started.

> *"This is one of the few Finnish demo parties where you can start drinking on the coach." -InvalidCo*

As the roadside dandelions and birch trees in spring bloom pass by, more and more free beer cans are opened and bottles of booze are passed around. This is not a quiet bus ride. There is lively chatter in both Finnish and English – some of the regular visitors come from Sweden, Norway, Germany and the Netherlands.

Organised travel

Kakka (Finnish for 'poop') is a familiar face in the Finnish demoscene and known for his quizzes; one is also held during this trip. Similarly to previous years, the prizes are booze and bad pornography.

"What else could a young man need?", as vELiKANi puts it.

As a woman, I let out a quiet sigh and decide to ignore the joke. In Finland, at least, the demoscene is a male-dominated culture, even though there are quite a few women. If your tolerance for such humour is low, the coming weekend may feel longer than it is.

After the quiz on Joroinen has been completed, the speakers on the coach start blasting Italo disco, a firm demoscene favourite. Some people get up

The Kakka quiz.

from their seats to socialise with the people sitting at the other end. Everyone seems to be in good spirits and most people have met previously.

"Simulaatios, or demo parties in general, are events for physical socialising. You get to meet people you won't get to see otherwise. Of course, there is the whole thing with the demoscene competitions, but for me personally it's less important than meeting friends", **ccr** tells me.

At the halfway mark, a quick stop is made at a service station and people stock up on drinks. Some people buy ice cream and eat them neatly outside. For the moment, the weather is sunny and mild.

After fifteen minutes, the group of people systematically board the coach – even when drunk, the group of demosceners is surprisingly disciplined. Even the driver complements us on how well organised we are!

Thoron, already very intoxicated, is firmly ordered off the tour guide's seat. VELiKANi tells people to ensure that **Astu**, known for heavy alcohol use, does not start drinking on board the coach.

For Thoron, the demoscene means "fun people, good feelings, entertainment, happiness, good friends, fun events, togetherness and art". I did not get a chance to interview Astu while it was still possible.

Joroinen is only 80 kilometres away. The last time we saw an urban landscape was in Vantaa; we are now in that part of Finland where you are more likely to meet a tractor than a hipster.

Those who order a shot called "Ritarin ryyppy" (Knight's drink) are knighted by the hotel owner Auvo. Naturally, a real sword is used in the ceremony.

As seen on TV

Some people have only bought entrance tickets, others have also reserved rooms. Those without a room can sleep on the floor. The entire hotel is reserved for the event.

"It's a special feeling when there are no outsiders around", says **InvalidCo**. Unlike at Assembly, for example, the entire event is about the demoscene, and its existence does not depend on the gaming convention that takes place at the same time.

The reception desk also works as a kiosk; you can drink tea and use the microwave oven for free.

When the coach arrives at the hotel, the parking lot already has groups of people drinking and smoking – bringing your own alcohol indoors is not allowed. The windows of the party hall are blacked out with plastic bags; inside, those who arrived on their own are already hacking away by the glimmer of the coloured spotlights.

The appearance of Hotel Joronjälki in the Finnish version of the TV show Hotel Hell put the spotlight on the hotel's peculiar architecture and interiors, as well as its eccentric owner **Auvo Puurtinen**. "Auvo from Joronjälki" is known as an art lover, and he has self-published a collection of love poems titled Sinulle, rakkaudella ("For you, with love").

It is difficult to say which one is more bizarre: Auvo or the hotel itself, which the host of the TV show, chef **Jyrki Sukula** compared to Twin Peaks, and Ilta-Sanomat, a Finnish tabloid, described it as "a messy mix of hotel, horror and a dumping site". The notion of a dumping site probably refers to the hotel's surroundings, where you can find a rusty bus, a motorboat and miscellaneous items from other more or less dead businesses that operated on the hotel's premises. This is just the right environment for a memorable demo party.

Adellan, who is experiencing Simulaatio and Joronjälki for the first time, describes her experiences as follows:

"Actually, I was expecting the worst possible scenario, that we would barely

have a roof over our heads. As a party, it's...surprisingly peaceful in my opinion."

Before the evening's programme starts, we still have time to visit the nearby Jari-Pekka shopping centre for a hamburger and some groceries. In smaller towns like Joroinen, shops tend to close in the early afternoon at weekends, so it feels unreal, yet very handy, to have a 24-hour grocery shop nearby. As an aside, the centre is owned by Pentecostalists and even mild alcoholic beverages are not sold, but there are a total of three different fast food restaurants.

Anything goes in the scene

In addition to the demos themselves, demo parties typically have competitions in other fields of digital art. The competitions are generally called "compos" for short.

The photo compo starts the show with entries of varying quality, as you would expect for a small party like this. Photos range from quick mobile shots to serious, professional photographs. The text mode compo takes us back to the roots of the demoscene, but the "short wild" gets us close to its very core – modern, even serious works of art that blend humour with politics and cringeworthy inside jokes. This subcategory of video art originally started as a technical competition, but took on a life of its own at the hands of different authors. The rules are relaxed and allow for many types of works, and the fact that the compo is not taken very seriously even inside the scene conveniently relieves the pressure on the short wild authors. This is probably the most experimental category.

"The demoscene is about creating things on a computer without commercial pressure, or any kind of other pressure. Very often, the spirit is similar to a game jam, and I think these two scenes are very close to one another. Game jams are about making something on the spot, whereas in the demoscene, you prepare something and finish it on the spot. If it's not ready, I mean", says **exca** from the group "wide load".

At the middle of the compo, Supadupa's black-and-white, abstract dancing lines set the bar higher for the later entries. This less experimental and more aesthetic piece is a comforting break amidst the restless, light-hearted videos. The next entry is something completely different.

Wamma's politically laden *War Against the World* starts off strong with a burning Finnish flag. The ISIS propaganda song "Saleel sawarim" plays in the background, although it takes a while to recognise it. "Suomi Viina" vodka, police cars, the Finnish police logo. The crown is removed from the coat of arms of Karelia. The camera zooms in on a red background and the red colour becomes liquid. The sword and crown are removed from the Finnish lion, and the same red liquid effect reappears.

At this point, the video is already past the agreed time limit, but it keeps rolling. The Tax Administration building and logo. A black Wamma flag, made to look like an ISIS flag, blows in the wind. A blue and white flag with a moon and star pattern is on fire. A man covering his face with a scarf waves a black vane in the backyard of a Finnish block of flats. At the end, a black flag flaps at the top of a pole.

> *"The demoscene is one fucking deranged environment and a dysfunctional family." -Kisu*

After the video ends, one of the organisers states that they had no time to preview the video since it was submitted at the last minute.

The video is a hot topic outside after the compos are over. It is clear that not everyone liked it. It will most likely be disqualified for exceeding the time limit, but the contents are also criticised. One of the participants feels that, while the video started well, the unrelated scenes at the end caused it to disintegrate. He emphasises that his feelings are not hurt; it was just that the humour at the end was in poor taste.

Later on, as we discuss something totally different, **drb** from Germany

tells me that you can be anyone in the demoscene, anything goes. Your views or political stances can be whatever you like. People are not limited. The works contain all sorts of commentary and contents, bad or good depending on who you are. At larger and more official demo parties, such as Assembly in Finland, suspicious material – such as swastikas or nudity – is not allowed.

The only real party

The dance music entries are heard after a short break. I mostly remember the names, which are very imaginative but seem to have little to do with the song itself. Although I am not a fan of music compos, the dance music compo is one of the highlights for **jaffa**, for example.

"My best Simulaatio memories are related to the dance music compo. All of a sudden, [in the early morning] on Saturday the dance music compo comes on, everyone is feeling great, and the party turns into a rave."

Cos, who took part in the music compo, says that Simulaatio is the last real demo party in Finland.

"Assembly does not count. I liked the Stream parties in Tampere a lot, but at the moment, this is the only real demoscene party we have."

"To me, the demoscene means creativity in general, a reason to make music or graphics or code for fun... And I must admit that it also signifies youth; when I was young, demos were the greatest thing I had ever seen on the screen. Nowadays, I must confess to being a bit of a retro geek. But still, I had an entry in the music compo this time; if I hadn't come, I wouldn't have written the song. You don't need to be good to participate, it's just important that you join in."

Every song attracts a couple of dancers in front of the screen. In the words of **Moptim**: you don't so much listen to the songs in the dance music compo, you feel them in your muscles. Every song also gets a round of applause from the crowd that has grown substantially thinner.

It has to be said that I have never seen young or nearly middle-aged men dance with each other completely uninhibited, like they do at a demo party. Then again, there are few places where you can dance in your bathrobe or completely naked, and dance moves where you flash your partner are not commonly allowed.

It should be noted to those unfamiliar with Finnish culture that, whereas in many other countries you would get thrown out and have the police called on you for such behaviour, in the context of Finnish sauna culture and demo parties this is considered normal, drunk fun.

The dance floor is not the only place with an intimate atmosphere; the lobby is also full of naked men who have stepped out of the downstairs bath barrel. At around one o'clock at night, alcohol-fuelled amicable relations between men can be witnessed in the front yard. Seeing your friends from far away during demo parties is an important part of the fun and a reason for visiting.

What makes sceners tick?

Hotel breakfast starts at ten during the demo party, in line with the sleeping rhythms of the guests. A jukebox plays Finnish 1980s rock'n'roll and many

guests take their morning coffee with a stronger side order. The breakfast table offers toast with different trimmings, canned yoghurt and boiled eggs. Coffee, tea bags, juice and water.

In the party hall, many are already busy coding, although most people are doing something else on their machines (modern laptops or 8-bit home computers), such as playing games or surfing the Web. The rainy weather has chased away the smokers and drinkers from the front yard. The bath barrel is empty, but the covered terrace offers a safe haven for nicotine addicts.

1980s music and 8-bit computers are very much tied to the demoscene, since a large part of it has to do with reminiscing about the machines of yesteryear.

"Naturally, we're looking back to our childhood after thirty years or so", **moonQ** confesses.

DrDoom from the Paranoids has been an active member of the demoscene since the 1990s. To him, the scene is an integral part of an important hobby – computers.

Back in the day, **Pyksy** was drawn into the demoscene by the amazement of "what people could get out of the hardware".

"On the surface, people compete against each other, but we're all still friends", he says.

The only row of windows that is not covered by plastic bags is letting sunlight into the far end of the hall. The sunlight allows us to admire the golden brown, figured velvet upholstered chairs and the burgundy red wall-to-wall carpet that is equally showy but

Grue is hosting a quiz where the participants try to guess the C64 game by its screenshot.

a complete stylistic mismatch. Dusty wreaths made of silk flowers, probably from China, decorate the walls. There is an uncannily realistic lump on the floor that has the figure of a sleeping person. Closer to the screen, we find more equally living things, such as taxidermied birds of prey and a deer.

I hear that there has been a fight last night and someone has thrown up. At two in the morning, people have been told to turn the music down "so as not to scare the neighbour's cows".

Compos and inside jokes

The loudspeaker informs that the bus excursion to the KGB Grill in Varkaus is departing soon. The Simulaatio parties were originally held in Varkaus, and the familiar eatery still receives a visit from the demosceners even though the party has moved 19 kilometres away to Joroinen.

It's 5 pm. The music compo starts with tiny music, where the contestants only have 128 kilobytes to create a pleasant listening experience. The rules for the listening music compo are more liberal; while it is going on, I head to the terrace to interview people. (My apologies to the participants.)

There is a group of nearly twenty people in the smoking area, although only some of them are smoking. Discussion is lively, and despite the cool weather, at least Thoron feels comfortable in speedos and is in a very good mood. This is the second time he tries to offer me a salty liquorice and vodka shot in a bottle cap, despite my numerous objections. Meanwhile, haohmaru offers me a stroopwafel, a syrup waffle imported from the Netherlands. It is much better received. Dutch demosceners and their stroopwafels seem to be inseparable.

> *"I was expecting the year's best demo party, even though it's not the biggest. I was not disappointed." -drb*

With the song "Ass N Titties" playing in the background, haohmaru tells me that to him, the demoscene is about people, creativity and drinking beer. The German **ohli** shares his view – but she also mentions demos. An anonymous respondent wants to add booze, drugs and hairy men.

The listening music compo is still going on as I return from the terrace. An entry from Sweden contains a credible rap part in Swedish. It reminds me that, while I may not have the energy to listen to all the songs, the compo is nevertheless taken very seriously by the participants and a lot of time is spent on the entries.

After six o'clock, it's time for oldschool graphics, newschool graphics and animated GIF. Finland is playing in the semi-finals for ice hockey at the same time, which affects the size of the audience, but the taxidermied animals fill in for the missing people. Nevertheless, the entries receive proper rounds of applause, and their standard is by no means affected by the small audience.

This year, newschool graphics has a clear winner, as there is only one entry. The animated GIF compo has the most participants; one of them is a beautiful rotating beer can in a C64 visual style. Two animations have clearly been made at the party place: one features a wobbly jello shot, the other has **tft** dancing in a

Haohmaru.

A member of Epoch reciting a poem generated by a neural network.

hotel room with his overpants around his ankles. Undeniably, the latter is very funny.

"Astu always loses his shoes at parties", I hear someone say as I leave for my hotel room after the compo. I would be lying if I told you this development surprises me.

I return in time for the poetry compo. This is a Simulaatio specialty, a "tradition" that was started last year in honour of Auvo. After five reciters, three more black horses enter the stage. The last one of them drops the microphone, breaking it. It's Astu.

At the end, there is an improvised poetry duel between the two winner candidates. It is between **stonda** and his poem *Homo homo thg* (reference to yet another inside joke) and the group Epoch, whose poem has been generated from the logs of the members' IRC channel with Markov chains.

It's man versus machine, and this time the jury decides in favour of man. Stonda wins a piece of firewood from the bath barrel that **nosfe** has used for writing his entry that was eliminated earlier. The real prize, however, is Auvo's self-published poetry book with a dedication.

The bath barrel is the reward

The main event is the demo compo, which draws a full crowd into the hall. There are only seats for those who reserved them on time; those in the back need to stand on their toes and still cannot see everything. Luckily, the events from the big screen are relayed live to the video stream on the Scenesat website and to televisions all around the party place. There might be room in the bath barrel for watching the compos. You can also watch the demos afterwards on your own computer, but for voting purposes, it would be nice to see them now.

Oldschool intro. This is the deep end of the demo scene.

In the age of Commodore and MS-DOS, the demoscene was mostly about who can do what with limited computing power and existing hardware. Later, with the development of technology, the artistic side of things has come into play even more. Both are still relevant, however.

"The demoscene means balancing between science and art", InvalidCo explains. "Demos are about trade-offs between technology and artistic impression."

According to **Grue**, the scene has changed since the early days.

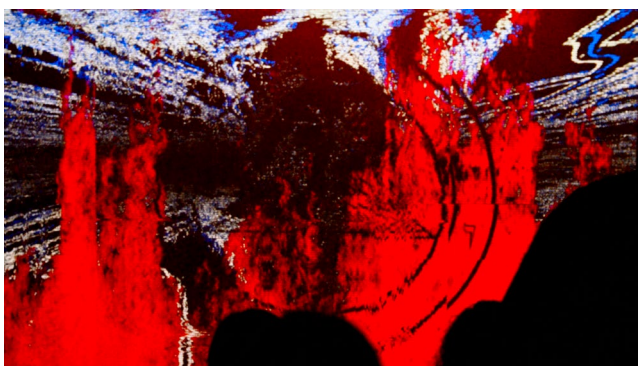
"The demoscene is about competition, testing your mettle. It's always great to see when someone exceeds the previous achievements. This is at the core of the demoscene. Nowadays, it's more and more about art, and I'm not really a fan of this. However, good code gets me all worked up!"

And good code is key when working on an old platform, such as the Commodore 64, PET or Amiga, and with size limitations; in the case of oldschool intro, the limit is 64 kilobytes.

The compo opens with *Ignition*, an abstract green and black beeper experience for the Commodore PET by the duo "oobc". I cannot recall the order of the rest of the entries, but my retinas recall visions of *Grizzly Bear* by Damones and *Greetings to Saunas* by BooZombies. *Vermeri tapaa asioita* ("Vermeri meets things") by ISO does not disappoint.

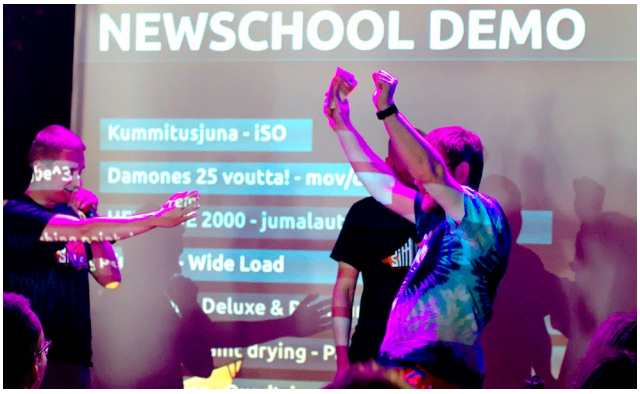
ISO is a Finnish demoscene phenomenon that I need to mention. It is a demo group whose members include nearly every active demoscener in Finland, unless they have somehow magically avoided it. There are no entry requirements and, therefore, no quality requirements. Naturally, this has consequences.

Since there may be up to one hundred ISO members, its releases are everywhere. A typical ISO demo is pure dada scribbled in MS Paint, with equally high-quality sound. As ISO, you can present a piece that would otherwise be embarrassing. People love



HELLRIDE 2000 by Jumalauta was third in the newschool demo compo.

Terwiz/Alumni's Three Shades of Gray for the 8-MHz 8086 Amstrad PC and a Hercules display.



urs celebrating the win of the newschool demo compo.


urs enjoying the bath barrel.

and hate ISO, and their productions usually elicit loud cheers. I feel like postulating that ISO in itself is a work of performance art that continues as ISO members release more work.

However, we need to go back to the demo compos.

The newschool intros are made with modern hardware, but they also have the same 64-kilobyte size limitation. Out of the four entries, the most striking is *Whitespace* by Prismbeings; it takes place in a dream-like white world of geometrical architectural models and Erik Satie's piano music in the background. There is some uncertainty regarding whether Prismbeings are allowed to use Satie's work, but demoscene productions rarely concern themselves with copyright matters. However, completely self-made productions are more highly valued.

This takes us to oldschool demo. The size of the demo is not limited, but the hardware must be manufactured prior to 1995.

ISO has three entries in this category; they are ranked third, fourth and fifth, with *Yleisradio Megademo* in last place. This is despite – or because of – the fact that the ISO demo "Mikon ihmissoppa" ("Mikko's human soup") needs to be shown at least three times, as it does not display correctly on the two first attempts.

On Sunday, we find that the winner is *Three Shades of Gray* by **Terwiz** (of Alumni), which also looks pleasing to my untrained eye with its rotating skulls made with character graphics. Second place goes to Ivory Labs' *Assembly 2016 Oldschool Demo Compo Invitation*, which is also the invitation to the oldschool compos at Assembly Summer. Invitations in entry form were also received from Chimpmembly and the Solskogen party in Norway.

Demos are not only works of art, in the same way they are not simply displays of technical skill. They can act as invitations, as is the case here, and including greetings to other demosceners and groups is very typical; this makes it even more difficult to compare demos to, say music videos or modern video art. Demos do not fit any other mould.

The finale for the compos is newschool demo, which offers eight audio-visual experiences, each one more beautiful than the last. Tomorrow morning, we will know that the winner is *Joroinen* by Deluxe and Premium, celebrating the very event where it was shown.

Once the demos are over, everyone can let loose, as long as they can somehow make it to the awards ceremony. The bath barrel and sauna are full of people, and the embers of the grill light up the brief period of darkness. Demos are projected on the back wall of the hotel, and some people are watching them from the comfort of the "demo vehicle", Terwiz's campervan.

Urs, a German living in Finland, tells me that making demos makes him happy.

"To me, the demoscene is primarily about making demos. The easiest way to be happy is to make demos. Well, maybe not the easiest... but anyway."

For urs, the bath barrel makes Simulaatio special.

"This party has a sauna and a bath barrel. Other parties only have a sauna."

The inner circle meets in Joroinen

The demoscene is behind many Finnish companies. **Mov** from Damones calls the demoscene an inner circle of cutting edge information – here, you will find the best experts.

Jaffa, a founder of many successful Finnish companies, has the demoscene to thank for many good things.

"My roots are in the demoscene, it led me to a lot of people and finally to the industry. All of my very important contacts were created through the demoscene. Without the scene, there would be no IRC-Galleria or any of the other companies I'm involved in."

"The demoscene is an active hobbyist community, and to me, it offered the chance to start creating products when I was young. When I was still in high school, I had already been to demo parties and made music and other things – I was already someone."

Jaffa says that Simulaatio is worth visiting.

"Simulaatio is a demo party that is very representative of the Finnish demo culture, and it's great to be here, far away. You see a lot of old friends and meet new ones. Simulaatio has a lot of inside stuff that only people who have been here will understand. Everyone should experience Simulaatio."

Links

- demoparty.net – Calendar of future demoscene events
- ftp.scene.org/pub/parties/2016/simulaatio16/ – Participating productions at Simulaatio
- www.pouet.net/party_results.php?which=1099&when=2016&font=none – Results from the compos at Simulaatio
- simulaatio.org – Event website

Researching the demoscene

Story by Markku Reunanen
Picture by Ji Hyun Hong

Despite the long history of the demoscene, academic research conducted on the topic remains scarce. Markku "Marq" Reunanen, who recently defended his thesis on the subject, tells Skrolli how he ended up studying demos and how it feels to put your own, beloved hobby under the microscope.

My first contact with the scene was by way of the crack intros in pirated Commodore 64 games in the mid-1980s. I saw intros at my friends' houses when gaming, but we were not interested in them; we simply wanted to play and quickly bypassed them by pressing the space bar.

My closest friend had an older acquaintance who supplied us with the best games of the time, first on cassettes and later on floppy disks. We were not concerned with the ethics of piracy, since everyone was doing it. Sometimes, the disks came with small programs that you could not play at all. These extras, which at the time felt fairly useless, played music or showed short animations.

I properly found out about the demoscene in the early 1990s, and soon, me and some like-minded friends founded a PC demo group called Fit and started to learn how to create code, graphics and music. It was difficult in the beginning, since we had to find out everything by ourselves. Much later, I found out that there were older Amiga guys in the same town who had been in some well-known groups.

From scener to researcher

My own demoscene hobby, which will soon be 30 years old, was naturally the basis for my doctoral research. Studying a subject you are intimately involved with has its pros and cons. On the positive side, of course, you save time: you already know the slang and conventions, and the demos you have seen have already given you an overall picture of the creative works of the scene. However, mere participation is not a research method in itself, unless you follow scientific criteria.

Whereas outsiders face the challenges of understanding the practices and unspoken codes of a community, those who study their group from within must be able to let go of their preconceptions. In other words, you need to be able to question what you "already know" and forget about your own likes and dislikes. Praising the scene and emphasising its uniqueness are potential hazards, but you must also avoid compensating for your background by underestimating its achievements.

Findings

Research concerning the scene has mainly focused on full-size demos. However, the community creates many other things: small demos known as intros, pictures, music, disk magazines, videos and even games. T-shirts, posters, badges and wristbands are among the non-digital artefacts from the scene. In order to bring out other types of works from among the shadows of demos, I focused on crack intros and 4k intros (maximum file size 4,096 bytes).

Hardware and operating systems are not simply tools; the scene has a much closer relationship with them. Advances in information technology and ever-changing market situations will unavoidably affect hobbyists, as their favourite devices become obsolete or marginalised. One of the most dramatic turns was the decline of Commodore, a 1980s favourite, in the beginning of the 1990s. Technological breakthroughs have been met with varying levels of enthusiasm: some have considered them to be the future of the community, while for others they have been its doom.

There are specific periods in the history of the demoscene during which like-minded hobbyists first gathered together and then started defining the habits and limits of their community. The piracy of the 1980s was closely connected to games, whereas gamers were clearly an outside group to the self-assertive scene of the 1990s. You can still feel traces of it at the yearly Assembly party, which became a disappointment for many purists due to the increased focus on games. *Kill all audio and lights!*

Times of Change in the Demoscene – A Creative Community and Its Relationship with Technology is available for download here: www.kameli.net/~marq/reunanen-times_of_change.pdf

A time machine for PC games

Paul Koller demakes modern games on the **Commodore 64**

A "demake" is a game that was originally created for a technically superior platform and then backported to older hardware. Paul Koller, from the Netherlands, specialises in demakes, and he has converted numerous indie favourites from PC to C64.

Story by Heikki Mustonen Images by Heikki Mustonen, Toni Kortelahti, Paul Koller

Creating new games for older hardware is nothing new in itself. Authors are drawn to retro platforms by the challenge brought about by limited memory and computing power as well as a certain fondness for a specific system.

However, **Paul Koller** is an exception in this group. His special expertise is converting fairly recent indie games to the Commodore 64. Some of these relatively new games focus on immediate playability and favour understated visuals. Nevertheless, squeezing even the smallest new PC game through this time machine is something you do not see every day.

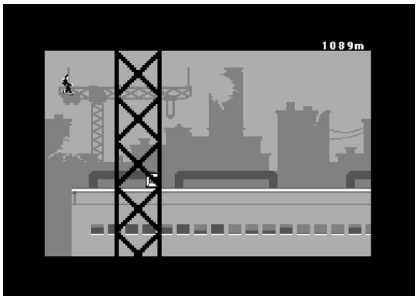
In 2016, the Koelnmesse conference centre in Cologne was full of Gamescom visitors. Skrolli met with the happy Mr Koller at the busy retro booth. While doing the rounds, he had seen at least two machines running C64anabalt, his Canabalt port from 2011. And his happiness was further boosted by the fact that a playable version of LuftrauserZ, a conversion of Luftrausers by the studio Vlambeer, was completed in time for the trade show. The gaming corner seemed like a good place to chat for a while.

A desire to work with C64 games often requires a long history with the machine. This is also the case with Paul, but one of my easy assumptions is wrong.

"Unfortunately, I was never a part of the demoscene. In the 1980s and 1990s, I did not have the necessary skills or contacts to be a part of it."

However, this did little to discourage Paul's interest in the machine over the years. The rise of indie games on the PC put the breadbox back on his desk.

"Interest in the conversions stemmed some 5–10 years ago. The indie games that came out then reminded me of the games I used to play as a child, but they had new and interesting game mechanics. Since I was still a C64 fan, I thought it would be a nice challenge to see if these new games could be adequately ported to old platforms."



The endless runner Canabalt, published for the PC and mobile devices, also runs great on the Commodore 64.

LuftrauserZ, a demake of Luftrausers, is Paul Koller's latest masterwork.

Every game is a challenge

Paul's background in technology and science encouraged him to take on the challenge.

"Today, game developers can focus on the mechanics themselves. They need not concern themselves with hardware limitations", Koller contemplates.

Overcoming technical challenges is a sufficient reward, since, according to Paul's calculations, the only development cost covered by the sales of these games is the electricity bill.

He had experience with several games before, starting with Luftrausers, but there are specific challenges related to each game. Beforehand, Paul anticipated that screen scrolling would be a big problem, but in the end it was fairly simple.

"It was a big help that I didn't need to scroll Colour RAM at all."

Collision detection, however, proved challenging.

"Creating the player's bullets was the largest technical challenge. They kept colliding with the other bullets. When using another weapon, I had problems with the edges of the screen and the game world. I thought I had corrected the problems multiple times, but they kept reappearing."

In the end, the testers finally gave the green light regarding the bullet problem, just in time for the trade show.

Koller has already created several conversions, but does the author have a personal favourite?

"Each game has a special feature, from a gameplay or technology point of view, that made it interesting to work on. Nevertheless, my proudest moment was Super Bread Box. It has the most content, and the conversion does not remove anything. All the fun and challenge is still included. Even after all these years, it's still a fun game to go back to."

Even though Canabalt and Micro-Hexagon are smaller games, Paul spots shortcomings in his conversions that the players do not seem to mind.

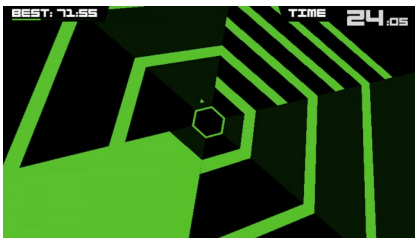
Paul has been working on LuftrauserZ for over two years now. Like many other conversions, LuftrauserZ is coming out on cartridge, but many matters related to the release were still incomplete at the time of the trade show. Decisions on the contents of the game package had not yet been made and the save routines related to the cartridge had not been programmed. Paul was hoping to have the game out by the end of 2016, but it did not happen.

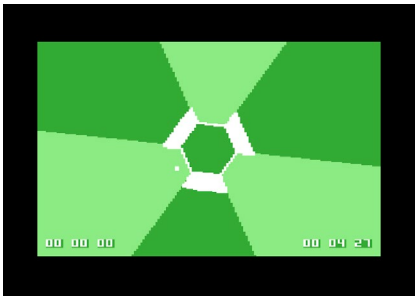
At time of writing, LuftrauserZ was slowly progressing towards publication. The scrolling text has been changed to "published in in 2017" and Paul confirms that the actual code is already done. "At the moment, we are finalising the cover art and waiting for Vlambeer's OK. This always takes more time than you would hope."

If you want to own a physical copy of LuftrauserZ, put your orders in. These limited editions always sell out without exception.

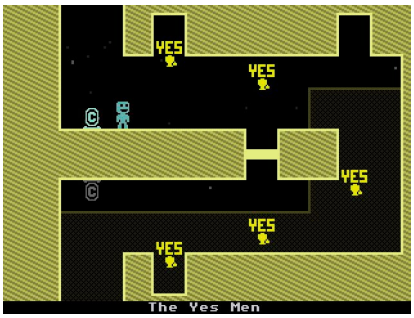
To play the games on an emulator or a real Commodore 64, you can download them from the C64 Scene Database, for example: csdb.dk/scener/?id=21728

More information on Twitter: @paulko64





Koller turned **Terry Cavanagh**'s Micro Hexagon into Super Hexagon.



The cult indie platformer VVVVVV also turned into an excellent game on the C64.

PAUL KOLLER'S DEMAKES

year	name	original	original developer
2010	VVVVVV demo	VVVVVV	Terry Cavanagh
2011	C64anabalt	Canabalt	Semi Secret Software
2012	Super Bread Box	Super Crate Box	Vlambeer
2013	Micro Hexagon	Super Hexagon	Terry Cavanagh
2017(?)	LuftrauserZ	Luftrausers	Vlambeer

CRATE VS BREAD
– how does a PC game run on C64?

Super Crate Box was the first game from the Dutch studio Vlambeer. The game came out in 2010 to general acclaim. At the moment, the action shooter is free to play on Steam, so it only costs a moment of your time to try it.

Paul Koller's conversion Super Bread Box came out two years later, with Vlambeer's blessing. Unfortunately, the cartridge version of the game was sold out some time ago, but you can download the game for free and play it on an emulator.

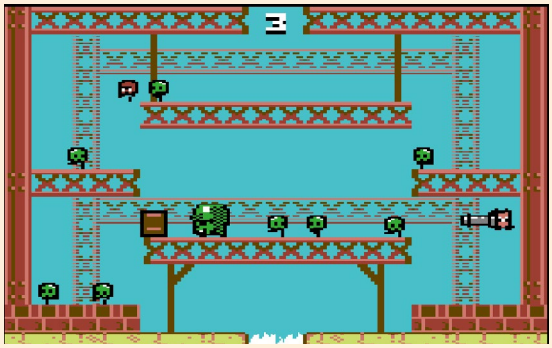
Super Bread Box is a pure one-screen shooter. The aim is to collect crates that give you one point each and switch your weapon. The enemies that proceed from top to bottom provide the challenge. A simple idea, but the controller gymnastics and swearing begin once you speed it up a bit.

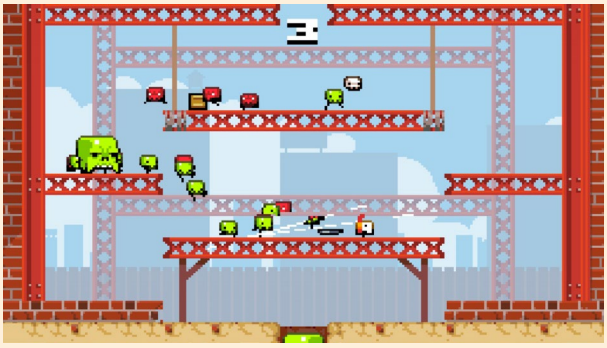
Vlambeer's original release already has a decidedly retro style, which has surely made Paul Koller's effort easier. The pixel graphics are clear and they have not overdone it with the colours or effects. Shooting adorable cephalopods is as fun to do as it is simple to follow. This is true for both versions of the game. The visuals of the conversion are even so good that you do not even miss the slightly better resolution and broader colour palette of the original.

When alternating between the different versions, you also notice how similar the controls are. The little butcher moves, jumps and shoots in a very similar manner in both games. A game's general tempo often speaks volumes about the precision and functionality of the controls. Super Bread Box hops around like the heart of a scared pig, and the C64 version does not fall behind. My gut feeling is that the game tempo is the same, which is a remarkable achievement for an 8-bit conversion.

If I had tried to guess in advance what the conversion will sacrifice, I would have been very wrong. Of course, you cannot fit every single thing in 8 bits, but the shortcomings are hard to find. The flame-thrower works differently, but there are no other apparent differences in the number and functionality of the weapons. Even the screen shake effects and minigun physics are included.

It is not difficult to see why Paul Koller is happy. Super Bread Box is an astonishing display of how graphics, music, the overall tempo and controls have all been carried over like carbon copies. The key component, the joy of gaming, is identical in both versions – as are the feelings of frustration and the "one more go" effect. An excellent duo that you should experience back to back. The order does not matter.





Koller considers Super Bread Box his most successful 8-bit demake.

FROM OUR GARAGE TO THE FINNISH MUSEUM OF GAMES

History in the making

The history of Finnish gaming had to be preserved. So we went ahead and did it!

Story by Mikko Heinonen Photos by Saana Säilynoja / Vapriikki image archive

For most of the international audience, the Finnish game industry brings to mind a few mobile gaming powerhouses like **Supercell** and **Rovio**, the triple-A releases from **Remedy** and a few smaller but successful developers like **Housemarque** and **Colossal Order**. Fewer people know, however, that the history of games published in Finnish goes back at least 155 years or that Finland's first commercial digital game was *Chesmac* from 1979, which was for the Telmac TMC-1800 computer that you had to build yourself. Despite this, it sold 104 copies.

All of the above are on display at the Finnish Museum of Games that opened at Museum Centre Vapriikki, Tampere, in January 2017. It has over 400 square metres of space dedicated to the presentation and preservation of digital and traditional Finnish games. On a typical day, the museum attracts over 1,000 visitors. However, its beginnings were humble and building it was a massive team effort.

Guilty as charged

It is hard to point out the exact moment when I started collecting. Maybe it was around the time when I bought a second-hand Commodore 64 for my brother to use in 1992. Nevertheless, during the 1990s, I found myself picking up old computers and game systems at flea markets and from friends. The only goal at that point was to try out everything I had missed out on in the 1980s – and, having exclusively owned Commodore systems up until then, there was a lot to catch up on.

Together with **Manu Pärssinen** (who does DTP for Skrolli) and my brother **Ville-Veikko Heinonen**, we launched the Pelikonepeijoonit website in 1999. What was originally intended as more or less a light-hearted joke soon became a central hub for the collecting of classic computers and videogame systems in Finland. We called it "The Arctic Computer and Console Museum" despite the fact that the systems were not on display anywhere outside of the images on the site. In fact, most of them were in piles of boxes in all of our parents' garages.

The website created some media interest, and we soon had requests to bring our systems to trade fairs and shorter exhibitions throughout the country. Over the next decade or so, we packed our things – first in vans, and then in lorries – countless times and set them up for people to play and admire. All the while, the dream of a real museum lived on.

Going academic

What had started as nostalgia-fuelled collecting slowly gave way to a more systematic approach to recording and preserving history. We started compiling facts about systems on our site and conducted original research. We helped our friend **Markku Reunanen** with his research and co-authored some papers on game preservation. I also completed a course in digital games design as part of my master's degree in English.

In 2008, the museum services of the City of Tampere were in a pinch. Their exhibition calendar had a 2-week gap, and since Ville-Veikko was working there at the time, the idea was put forth that Pelikonepeijoonit could set something up. The first ever exhibition bearing our name was a resounding success, attracting thousands of visitors and massive media attention in a short time. It was the start of a cooperation that would eventually lead to the creation of the museum.

The next milestone came in 2012, when the Game Research Lab of Tampere was setting up an exhibition about Finnish games for a seminar. Again, the city's museum services and researcher **Outi Penninkangas** were involved. While the exhibition "Finnish Games Then and Now" only lasted for a week, it introduced us to **Annakaisa Kultima** and her students and colleagues who enlisted our help with the hardware and exhibits. We now knew that the local university also had people interested in the preservation of gaming culture, and this would be pivotal in the planning and promotion of the next project.

Crowdfunding picks up

2012 was also the year of Kickstarter and the massively funded retro game remakes. I was

watching the scene closely and wondering whether something similar could be done in Finland. The local legislation has some quirks that prevented a straight-up fundraiser, but I played around with the idea of selling advance tickets. I contacted the museum about a potential crowdfunded exhibition, but the idea was not mature enough. I left it on the backburner for the winter.

The next year, a large computer and electronics retailer set up an exhibition of computers and video game systems at one of their stores. It was, and still is, a nice view into the history of hardware, but Pelikonepeijoonit had always been more about playing than simply *displaying*. We wanted something interactive. So when Manu and I met **Tuija Lindén**, the Grand Old Lady of Finnish game journalism, at an event arranged by *Trials* developer **RedLynx** in Helsinki in the spring 2014, the stars were finally aligned. She wanted to know if we could finally build a museum for games. I told her we would give it our best shot, and that it would be built in Tampere, since I already knew all the right people there.

After the conversation with Tuija, I knew exactly what we had to do. I contacted Outi, Annakaisa and a few other people and introduced the idea of a crowdfunded museum. Much to my surprise, they ate it up. The museum already had a space we could use and even the deputy mayor of the city turned out to be an avid NES fan. He arranged for me to pitch the idea to the regional development company, and it was the easiest money I have ever raised; they simply loved the idea and would absolutely back it. In exchange, the museum would cough up a sizeable sum themselves. We would have an exhibition in place even without the crowdfunding.

However, it was important that we involve the community, and I felt that crowdfunding would be the best and most cost-effective way to do it. Nothing like this had ever been attempted before in Finland; in fact, no government entity had ever done crowdfunding. I was amazed when I heard that the City of Tampere was willing to attempt it – the finance people and lawyers could have easily said no, but instead they wanted to see if it was possible. It was.

Six months of trials

Everything up to this point had been plain sailing, but now we were facing the great unknown. Our target of €50,000 was high, clearly above the previous crowdfunding record on the Finnish **Mesenaatti** platform (which we had to use due to legal reasons). Luckily, our core team was reinforced by fellow Skrolli contributor and veteran journalist **Jukka O. Kauppinen** and researcher **Niklas Nylund** from the Rupriikki Media Museum. We spent weeks fine-tuning the campaign page, perks, stretch goals and video before finally launching the campaign on 31 March 2015. We had chosen an exceptionally long campaign period of 6 months, as we wanted to get past Finland's famous June–July shutdown.

To say that the following months were intense would be an understatement. We updated our Facebook, contacted potential backers, did dozens of interviews in the media and tried to keep calm during those inevitable days when we had absolutely no sales. However, things started to really pick up towards the end of August, when we were able to secure major backing from Supercell, Housemarque and Colossal Order. In the end, we doubled Finland's previous crowdfunding record with €85,860. Since then, it has only been beaten by another museum that used our campaign template – and they only raised some €500 more.

SUOMEN PELIMUSEO

Dream come true

The successful crowdfunding allowed us to make the museum an interactive experience. There are currently 100 Finnish games from 1862–2015 on display, and over 60 of them are playable. Furthermore, the museum contains a video arcade with about a dozen classic coin-operated machines for the visitors to try. One of the stretch goals of the crowdfunding was a Twilight Zone pinball machine, which has proved immensely popular: during the first three months of the museum being open, it racked up over 15,000 games.

There are projects where it seems that nothing goes right, and ones that succeed despite great adversity. While building the museum was by no means a picnic, there were never any moments of deep despair. We all knew we had the right idea, and everyone we spoke to supported us either financially or at least spiritually. The entire Finnish gaming industry came together and helped us build the exhibition, donating and loaning items that would otherwise have been unobtainable.

By the end of June 2017, over 100,000 people had already visited the museum. To me, this is nothing short of a dream come true and further proof that games – both traditional and digital – have become a generally accepted part of culture and worthy of preservation. The time to build a museum for them was now, and so we went ahead and did it!

Sacred geometry
– The source code of our reality?

We published this story in our Finnish magazine when the project it discusses was free to use. It has since become commercial, but the story is so interesting that we wanted to show it to you anyway. This is not an endorsement of the commercial product.

Teksti: Sakari Lehtonen
Kuvat: Sakari Lehtonen, Randy Son Of Robert (Flickr), ESA & NASA

In the early 2010s, I became interested in the significance of text symbols, particularly numbers. I had learned how to communicate, calculate, write and program computers with the Western symbols we are all familiar with.

I am an engineer and a self-taught software developer. It felt silly to understand the workings of a processor on a binary level without understanding how language works beyond words and numbers.

As a programmer, I had learned to use text and numbers very effectively as a means of expression.

Despite this, I could not even grasp the basics of the logic of how glyphs and numbers represent the meanings we give them. Why do even these letters and words come together to form concepts that become concrete structures and ideas in our thinking?

Similarly to how computers have higher and lower level languages, it occurred to me that perhaps the human language has lower levels that have been buried by a high level of abstraction.

I started to explore the meanings of the building blocks of written and oral communication, i.e. letters and – in particular – numbers, when they are studied as individual entities.

I had always wanted to understand matters deeply, such as the low-level operation of a processor. This question started to greatly fascinate me.

I wanted to understand the deepest meaning of symbols: to disassemble their visual structures into the smallest parts and to see what they stand for. To understand why specific shapes represent specific phonemes or amounts.

The geometric nature of numbers

I started to examine the matter from the point of view of numbers, since there are fewer numbers than letters and I already understood their behaviour better. Mathematics provided a clear set of rules for understanding their relationships.

I started to examine numbers and the amounts they represent in a visual form in order to better understand the relationships between them. To me, the simplest and most natural approach seemed to be shapes based on circles, with the points along the circumference representing the amounts.

I first drew patterns based on circles by hand, using a pencil and a pair of compasses. Individual numbers were represented by individual circles, while their repetition at different points of the circumference represented the multipliers.

I discovered that this is too labour-intensive. I tried to do the same with traditional image processing software, such as Photoshop and Adobe Illustrator. This also proved too laborious due to the amount of repetition required. Automation was needed in order to achieve the desired results.

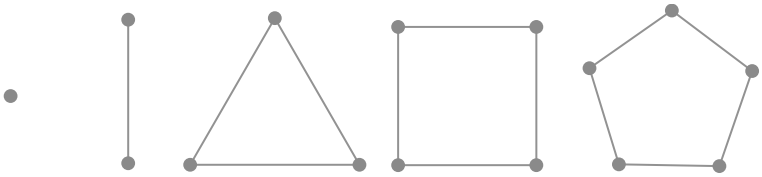


Image 1. A visualisation of the numbers 1–5.

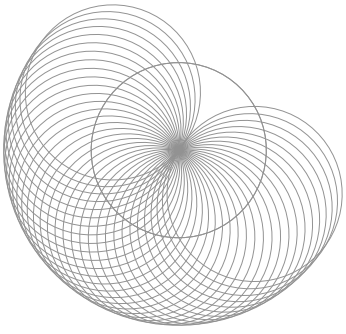


Image 2. Repetition of a circle along the circumference.

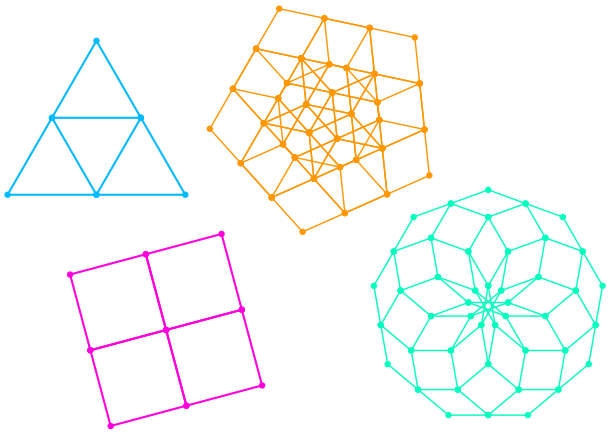


Image 3. Different polygons.

Visual programming

I decided to try to program these patterns. Initially, I created a simple program with the Processing.js library. It allowed the user to interact with the polygons shown on the browser screen.

I could add circle-based shapes on the screen and edit them. Furthermore, I was able to increase the number of points, increase recursion, scale, copy and modify other pattern parameters. Naturally, three was expressed as a triangle, five as a pentagon and so forth.

Combining basic shapes with fractals, recursively repeating patterns, created interesting and exciting pictures. Many of them were reminiscent of a digital flower, something living and beautiful.

I found traditional, Mandelbrot-type fractals boring since they were so predictable and flawless. With this software, I could create recursive geometry that contained mistakes, exceptions from the rule. These flaws made the patterns more natural, soulful and lifelike.

Playing with patterns

Using this simple software felt like a natural way to process numbers and their amounts within my conceptual field. I continued developing the software while examining the relationships between numbers through play and the discovery of new patterns.

This software allowed me to create abstract, geometry-based art in an interactive manner. It felt personal and meaningful. I was expressing myself by using a very rational tool, while utilising my creativity in the process at the same time.

This seemed to activate both my artistic and logical-rational thinking, bringing them closer together. In a way, it created a bridge between these two sides – as if they were now able to exchange ideas better despite the traditional separation.

This felt good. The function satisfied both my logical side, which understands numbers and enjoys organising things, and my more creative, artistic side.

Creating geometry seemed to open up my thinking and helped to combine different, possibly unrelated matters within my conceptual field and to see complexes from different angles. This helped to bring thoughts from the logical side to the creative side, and vice versa.

This is nothing new; the same can be observed by creating art. Thoughts pass elsewhere and the artist can focus on things they would not otherwise consider. However, these very specific patterns and their rules seemed to penetrate deep into my being – in a very significant manner that I had never experienced before.

Sacred and natural geometry

I continued developing the software, and further investigation revealed that art such as this has been found everywhere in the world. In the Western world, this art was commonly known as "sacred geometry". In this context, "sacred" refers to the use of specific proportions in churches, temples, mosques, pyramids and other spiritual locations around the world.

Almost without exception, the ratio occurring in these buildings and their architecture, art and structures is the golden ratio, Phi (φ), with an approximate value of 1.618.

Phi is a ratio that can be found to reoccur in nature; in flower petals, tree branches and cone shells.

I copied these patterns into my software. I favoured the golden ratio as a parameter in the software, since I found that it produces beautiful pat-

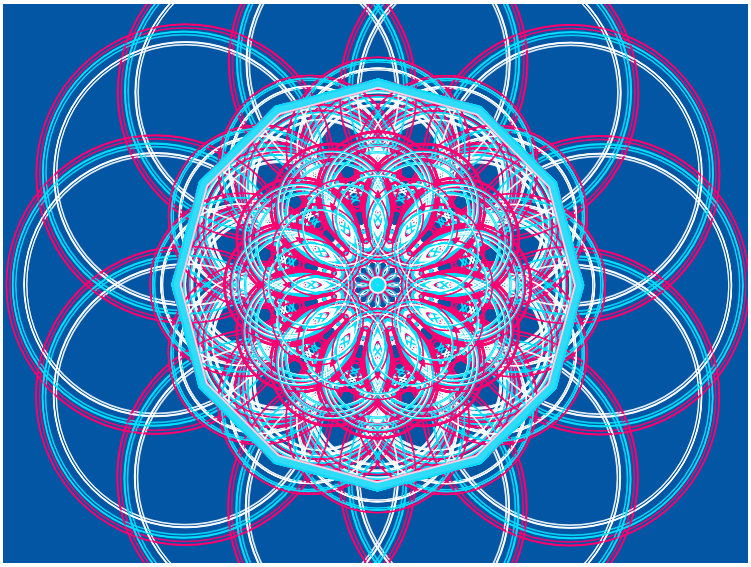


Image 4. Digital flower.

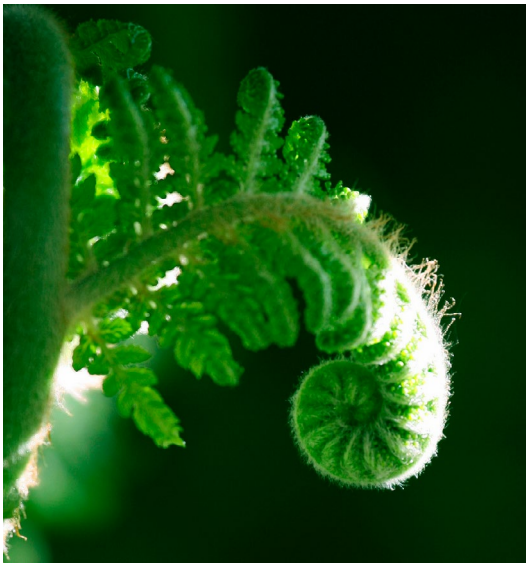


Image 5. Golden ratio in plants.

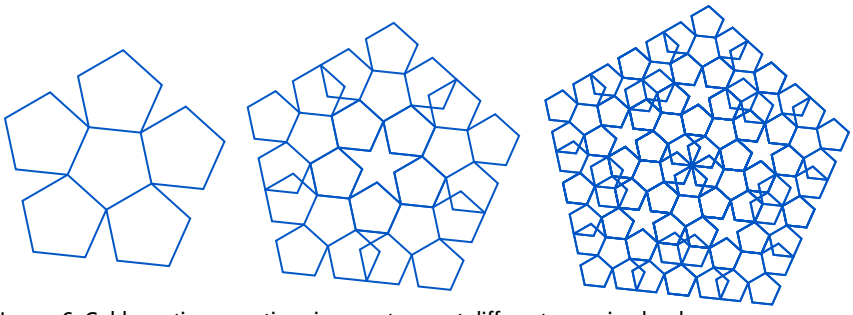


Image 6. Golden ratio proportions in a pentagon at different recursion levels.

terns and structures. When using the golden ratio, completely new and unexpected patterns formed at the interference points of the patterns; see Image 6, for example.

With these default templates, I quickly started to achieve beautiful results and found that playing with the software was a lot of fun! I gained pleasure from creating new versions of the same basic pattern – simply by changing one of the dozens of parameters that were used to create the patterns.

The pleasure of creation

Modifying, copying and colouring shapes with the keyboard and mouse felt good – it clearly activated the pleasure centres in my brain.

At some point, I noticed that this may actually result in changes in my thinking. By creating a beautiful basic pattern while changing the level of recursion and the number of points, I started to see new, unexpected patterns at the intersections. For example, the monk character created by the combination of the lines that I found completely at random while experimenting with different combinations of seven dots.

Discovering new patterns created a rewarding feeling that was comparable to finding something beautiful and unexpected in nature. It was as if my mind had given me a gift after I had fed it properly.

With my software, I was able to repeat this feeling time and time again, finding new patterns in the same basic structure by slightly altering the parameters. Once I had found something interesting, I modified the number of points, for example; this caused the same pattern to repeat at higher and lower point counts, but from a completely different angle.

This gave me continuous pleasure in discovery and seemed to activate those parts of my brain that respond to good music or the type of humour where you expect something to happen next but it comes from a different angle – as a variation you did not expect.

Finding life in the patterns

The proportions of the geometry I had created were such that they became linked to life in my mind; I started drawing comparisons with the facial features of animals or the proportions of a human face, such as where the eyes are located in proportion to the nose and mouth, or with different flowers and other plants.

Dusty Conley, a user of my software, demonstrates this phenomenon by painting animals on top of the discovered patterns, such as the tiger in Image 9.

In a geometric trance

Working on these patterns for several hours could even bring me into a trance. One time, after rising up from my desk, I felt as if my entire being was focused on one point. I was more focused than nearly ever before in my life. My entire field of vision seemed distorted to the point where I could only feel myself and this moment, this point in time. The lights around me were glowing in a peculiar manner, and it was as if my entire being had resonated at a high frequency for dozens of minutes after using the software.

When going to sleep after this experience, I had the most powerful lucid dreams I have ever experienced. In my dream, I experienced that the entire structure of the universe consists of small particles that repeat themselves in a simple manner, combine, divide, proliferate and form ever more complex structures, constantly repeating and following the same basic rules.

When I woke up, I was very confused regarding what was happening. I never expected to reach these states of mind by creating geometry with my software.

Many times, the patterns I had created continued to live on under my eyelids. They might continue developing within my field of vision for a long time after I had stopped the active work. The pattern I had been working

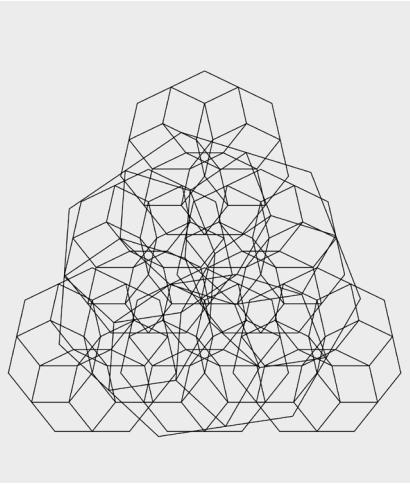


Image 7. The interference reveals a sitting monk.

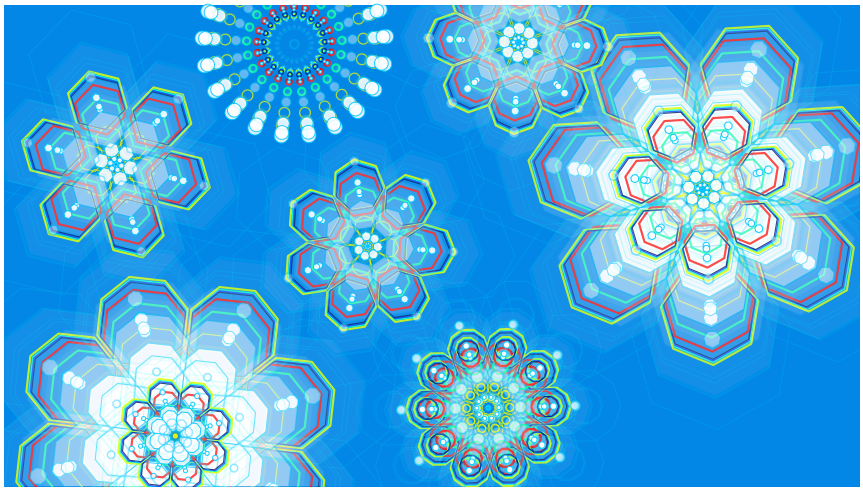


Image 8. Different variations of the same formation.

on could gradually develop into something completely different – something I could not even have imagined within my field of vision.

Sometimes, the patterns lived on for several hours. When I closed my eyes or stared at a wall, I could see the pattern as a weak, transparent layer, repeating itself, developing and living, breathing in new forms and structures.

More and more, I observed that creating the patterns affects my thinking in a unifying manner. I could build bridges between different things that I had previously been unable to combine.

These experiences fuelled my development work. I wanted others to be able to have similar experiences. I called the software GeoKone and published its first public and free to use version on 11 November 2011.

User experiences

After publishing GeoKone, I started receiving feedback. The most enthusiastic users reported similar, odd feelings and thoughts:

"The ideas of macro and micro worlds inspired by Einstein and his unified field theory: 'The infinity of data, both inward and outward', from atoms to universes and all their relationships. GeoKone helped me to bring all these thoughts to the visual reality. Understanding this was life-changing to me to say the least."

The feedback I received emphasises the feeling that the creation of such geometry may set in motion something very powerful and meaningful inside us.

The source code of our universe?

I have only scratched the surface of the deep meanings of numbers and letter symbols. However, the years I have spent researching the topic, talking with people and exchanging experiences have strengthened my understanding of the fact that there exists a language of geometry that reflects our reality and the structure of our inner selves.

By connecting with these fundamentals through a creation process, we can understand ourselves as larger parts of these structures at a very fundamental level.

According to quantum physics theories, things only exist when we examine them. Could this also work the other way round – where we, in fact, create everything we observe?

Perhaps, while actively creating geometry that follows nature's rules of creating we are redefining the connections inside us that define how we transmit information at the cellular level. This could be an explanation for the mystical experiences that I and other users of my software have encountered.

Visions of healing software

I believe that exposure to correct visual shapes may have a unifying, holistic effect on us. The healing power of music has already been studied and it has been shown to be powerful. In the future, similar effects may be discovered in carefully adjusted visual shapes.

I am continuing my work with this subject and developing software that aims to positively affect the user's state of consciousness by combining geometry and sound in meaningful ways.

As a nascent technology, virtual reality allows for taking this experience to an entirely new level, especially when the user feels that they are creating the world they are involved in. Based on this idea, we have started a project called Geometrify (geometrify.net). I have spent the last two years developing my own 3D engine (C++, Lua and OpenGL) for Geometrify. I wanted to deeply understand how to generate graphics that affect the consciousness. I also wanted to make the most of the hardware, and the intention is to utilise the software at many other locations with a view to the long-term vision.

We have shown a demo of Geometrify at over 25 public events and heard of amazing experiences from its users. We have offered over one thousand people a sample of this world, while also providing them with their first VR experience.

Image 9. A tiger born from geometry.

Image 10. The galaxy.

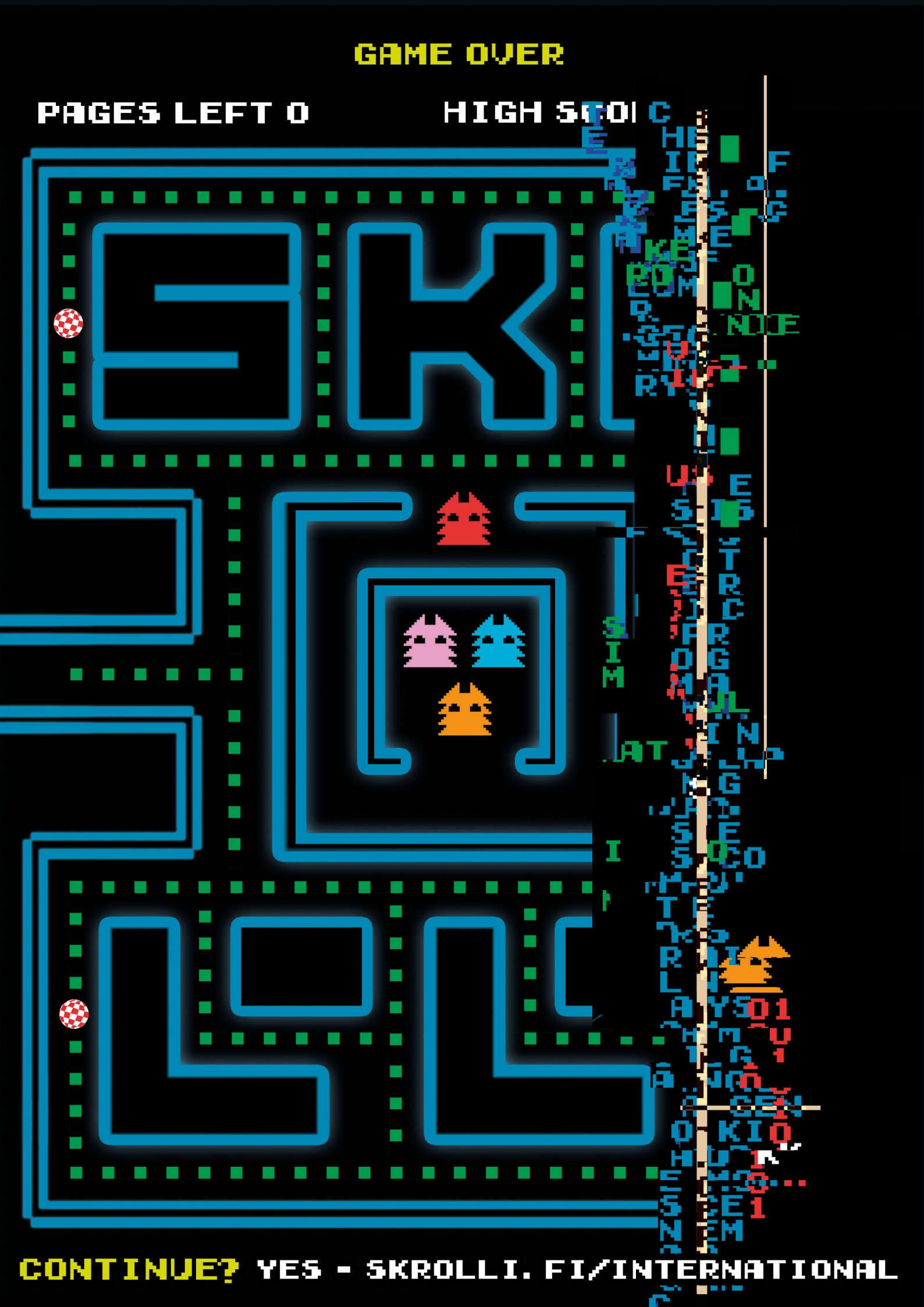
Create geometry yourself

I recommend that you create sacred geometry yourself. The concept is difficult to grasp by reading or looking at pictures. By drawing or colouring these shapes, you can also learn practical trigonometry – by repeatedly drawing triangles with the golden ratio, you can store their proportions in muscle memory. **Miika Kuisma** has used GeoKone to produce colouring templates that you can print out. They are available at naturalsymmetrycoloring.tumblr.com.

Later, I transformed GeoKone into a commercial application called Omni-Geometry. I felt a sense of accomplishment in being able to turn a project that started as a personal experiment into something much bigger in the end.

GAME OVER

PAGES LEFT 0 HIGH SCORE



CONTINUE? YES - SKROLLI.FI/INTERNATIONAL